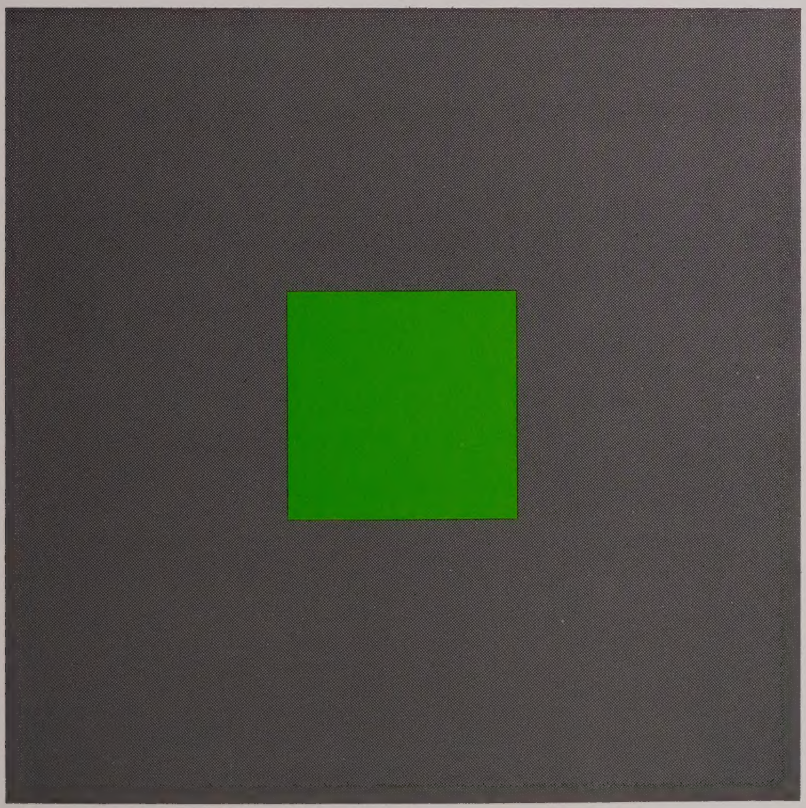
ISSN
0955-8543

HYPERMEDIA



Volume 2 Number 1 1990

TAYLOR GRAHAM

Editor:

Patricia Baird, Department of Information Science, University of Strathclyde, Livingstone Tower, 26 Richmond Street, Glasgow G1 1XH, UK.

Associate Editor:

Noreen Mac Morrow, Department of Information Science, University of Strathclyde, Livingstone Tower, 26 Richmond Street, Glasgow G1 1XH, UK.

Editorial Board:

Robert Akscyn, Knowledge Systems Incorporated, 4750 Old William Penn Highway, Murrysville, PA 15668, USA.

Professor **Peter Brown,** Computing Laboratory, University of Kent at Canterbury, Canterbury CT2 7NF.

Mercedes Caridad, Sociedad Espanola de Documentacion e Informacion Cientifica, Gran Via 67, Despacho 302, 28013 Madrid, Spain.

Phil Chauveau, Managing Director, Apple Computer U.K. Ltd., Hemel Hempstead HP2 7HQ, UK.

Professor **Blaise Cronin,** Department of Information Science, University of Strathclyde, Livingstone Tower, 26 Richmond Street, Glasgow G1 1XH, UK.

Elizabeth Duncan, University Teaching Centre, University of Aberdeen, King's College, Aberdeen AB9 2UB, UK.

Dr. **J.T. Mayes,** Scottish HCI Centre, George House, North Hanover Street, Glasgow G1, UK.

Barry McIntyre, Department of Information Services, Royal Melbourne Institute of Technology, GPO Box 2476V, Melbourne, Victoria 3001, Australia.

Dr. **Cliff McKnight,** HUSAT Research Centre, University of Loughborough, The Elms, Elms Grove, Loughborough LE11 1RG, UK.

Professor **Ted Nelson,** Project Xanadu, 2320 Marinship Way, Sausalito, California 94965, USA.

Dr. **Jakob Nielsen,** Department of Computer Science, Technical University of Denmark, Building 344, DK-2800 Lyngby, Copenhagen, Denmark.

Professor **Roy Rada,** Department of Computer Science, Liverpool University, P.O. Box 147, Liverpool L69 3BX, UK.

Dr. **David Raitt,** Frankenslag 179, 2582 The Hague, The Netherlands.

David Riddle, King's College London, Educational Computing Unit, Centre for Educational Studies, 552 King's Road, London SW10 0UA, UK.

Iain Ritchie, Managing Director, Office Workstations Ltd., Rosebank House, 144 Broughton Road, Edinburgh EH7 4LE.

Professor **Ben Shneiderman,** Department of Computer Science, University of Maryland, A.V. Williams Building, College Park, MD 20742, USA.

Dr Dr **Norbert A. Streitz,** GMD, Postfach 104326, Dolivostrasse 15, D-6100 Darmstadt, West Germany.

Dr **Randall Trigg,** Institute for Information and Media Sciences, Niels Juelsgade 84, DK-8200 Aarhus N, Denmark.

Nicole Yankelovich, Institute for Research in Information and Scholarship (IRIS), Box 1946, Brown University, Providence RI 02912, USA.

Subscription Information

Hypermedia is published in Spring, Summer and Winter. Subscription price for Volume 2, 1990, to include postage and handling: £49.00/US$92.00. Copies are sent by surface mail. All subscription orders, back issues, advertising and other enquiries should be sent to: Taylor Graham Publishing, 500 Chesham House, 150 Regent Street, London W1R 5FA, UK.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or be transmitted in any form or by any means (electronic, mechanical, photocopying, recording or otherwise) without the written permission of the publisher.

© 1990 Taylor Graham and contributors.

Hypermedia
VOLUME 2 NUMBER 1 1990

INCREASING THE POWER OF HYPERTEXT SEARCH WITH RELATIONAL QUERIES

LEONARD GALLAGHER, RICHARD FURUTA *and*
P. DAVID STOTTS
National Computer Systems Laboratory,
National Institute of Standards and Technology,
Gaithersburg, MD 20899
and
Department of Computer Science, University of Maryland,
College Park, MD 20742

We describe an SQL relational database schema for representing the objects in HyperCard, along with a technique for automatically populating this schema from a HyperCard stack using the facilities in HyperTalk with calls to the database manager. The standard relational database query language SQL can then be used to perform more general hypertext searches than are possible with the string search feature found in most hypertext browsing environments. Semiautomatic updates of the content of a hypertext are also possible using SQL updates on the object representations in the database to trigger corresponding HyperCard updates on the objects themselves. We describe a prototype implementation and present several example queries and updates to motivate this approach. These techniques, although demonstrated here specifically using HyperCard and Oracle for Macintosh, are generally applicable to a wide range of hypertext systems and relational databases.

1. INTRODUCTION

Location of information elements in a hypertext by means other than browsing is an important problem. In a very large document, a fast searching facility can provide starting points for further browsing. This saves the time required to browse to each starting point, as well as helps to prevent reader confusion and potential loss of context from viewing irrelevant information along the way.

Searching in hypertext systems is commonly done via a textual mechanism in which the content of an information element is examined for some target string of characters. In some systems, for example Shneiderman's Hyperties[10,8] string searching is made a bit more specific by designation of fields in an element to which a search may be restricted, such as 'author', 'title', or 'abstract'. In addition, most string search mechanisms include some logical operators for composing longer queries from simple ones.

String search, however, is limited in its power because it can only locate elements according to their textual content. As a part of the Hypertext Competency project at the National Institute of Standards and Technology, and in conjunction with the Trellis project[11] at the University of Maryland, we are examining a more powerful hypertext searching facility using a relational query language and an automatically-derived relational database. The increased power of this approach over string searching alone is that logical and physical properties of an element and its presentation can guide a search. In addition to increased searching power, a database approach also allows updates to the information content of a hypertext through common relational query language features.

In this report we discuss one particular implementation using the HyperCard hypertext system and the Oracle relational database system. As a default minimum, the schema in our implementation captures the physical properties of each component of a HyperCard stack, properties that should be of general applicability to all HyperCard users. An author can employ the default schema as it is, or extend it with other information representing logical document properties of specific interest to him. Finally, the schema is automatically populated for a stack by execution of a HyperTalk[4] script after the stack is written. Extensions of this concept are under development to employ an author's think time by populating the schema as a hypertext is written.

The combination of hypertext systems with database technology is not a novel development *per se.* Hypertext can be used as a front-end database mechanism to ease the difficulty of query formation. Delisle and Schwartz, for example, have developed a hypertextual query interface for a CAD database.[6] The Oracle company has produced a similar system for general applications, using HyperCard as the query-construction and result-viewing mechanism for SQL databases. On the other hand, hypertext systems can be implemented using relational databases instead of explicit linkage representation. Scacchi, for example, has constructed a hypertextual software engineering environment[9] on top of the UNIX Ingres database system. Our project offers a third approach to joining databases and hypertext. We essentially are borrowing the searching and update power of the relational data model and applying it to the network data model (which is more useful in information domains composed of graphics and large chunks of text). Cooperation is accomplished by automatic conversion of information from network form to relational form, and back again.

In Sections 2 and 3 we present the logical structure of our prototype

hypertext/database combination. Section 4 then provides several sample queries in the SQL language to illustrate the power and utility of relational search and update in the hypertext domain. The report concludes in Section 5 with a discussion of the applicability of these principles to other hypertext systems and relational databases.

2. THE HYPERCARD DATABASE SCHEMA

HyperCard[3] is a programmable hypertext system for the Apple Macintosh that incorporates five major physical features: CARD, BUTTON, FIELD, BACKGROUND, and STACK. Cards are the basic containers of information that act as screens for graphics images and as repositories for buttons and fields. Buttons are 'hot spots' on the card for linking or sending messages to other objects and fields are 'containers' for other information, usually text. A stack is an ordered list of cards that supports 'next' and 'prev' operators so that all cards in the stack are sequentially linked. Backgrounds are special cards that partition the (foreground) cards and support images, buttons, and fields common to each group. Buttons or fields on a background card are automatically available to all foreground cards sharing that background.

We model the HyperCard physical objects with five relational database tables, one for each object. Each table consists of a primary key identifier for the object as well as attribute fields to fully describe its display characteristics and size, position, appearance, and maintenance properties.

Each instance of a HyperCard object may have an associated script written in HyperTalk, the programming language for manipulating stacks[4]. A script enables an object to send and receive messages. It is a collection of 'handles', or message handlers, each triggered when a matching message is captured by the object owning the script. Once triggered, the HyperTalk commands, functions, and control structures defined by the handle are executed. Any 'GoTo' statement in a script defines a link going from the underlying card of that handle to the target card of the 'GoTo' statement.

We model the message passing and card inter-connection properties of HyperCard with two additional relational tables, HANDLE and LINK. The handle table identifies each message handler by its name and by the physical object in which it resides; it also stores length and complexity properties of the identified handle. The link table represents each link as an ordered pair of card identifiers, source and target. The link table also identifies the originating handle and indicates whether the link is 'direct', i.e., always

followed when the handle is triggered, or 'indirect,' i.e., contained in some conditional structure that may or may not execute the 'GoTo' constituting the link.

The relational schema we use to model HyperCard documents thus consists of the following seven relational tables:

```
CARD (card_id, stack_id, bkgrnd_id, card_properties ...)
BUTTON (button_id, card_id, stack_id, button_properties ...)
FIELD (field_id, card_id, stack_id, field_properties ...)
BACKGROUND (bkgrnd_id, stack_id, bkgrnd_properties ...)
STACK (stack_id, stack_properties ...)
HANDLE (handle_name, card_id, stack_id, contain_obj_id, ...)
LINK (source_card, target_card, handle_id, ...)
```

The full SQL schema is included in Appendix A. This basic schema captures the minimal physical and logical properties of a HyperCard document and contains sufficient structure and organisation information to answer many questions not based on the textual content of fields. A hypertext author could extend this schema to include usage-specific properties of particular interest. For example, all fields of a document could be tagged with a 'security level' identifier that gets stored into the FIELD base table, or, in a hypertext containing geographic maps, each row in the CARD table could identify geographic conventions specific to cartography.

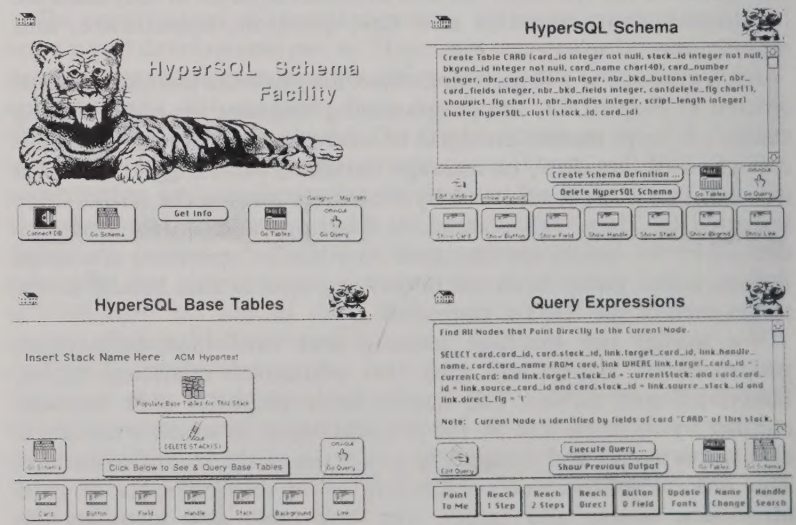


FIG. 1. *Main cards for database construction and querying*

3. AUTOMATIC SCHEMA POPULATION

Our prototype implementation, the HyperSQL Schema Facility, is a tool intended primarily for hypertext authors to use during document development. It provides functions for creating and populating the relational schema described above, for querying the resulting SQL database, for selectively browsing the represented stacks, and for cascading modifications of the database to the HyperCard stacks themselves. The complete facility consists of three logical parts: schema maintenance, base table maintenance, and data manipulation. Fig. 1 shows screens from each component of this implementation.

Schema maintenance provides SQL definitions of the seven schema base tables and allows user modifications or extensions to capture additional meta information. The main objective of the 'HyperSQL Schema' card shown in Fig. 1 is to create the SQL database definitions that will represent HyperCard documents. After the (seven) table definitions have been suitably prepared, the 'Create Schema Definitions . . .' button sends them to the underlying SQL database management system, which in this case is Oracle for Macintosh.[5]

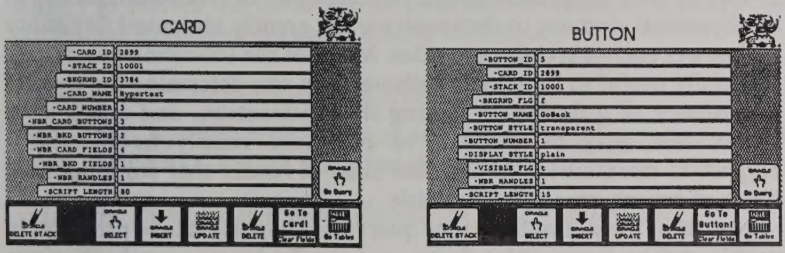


FIG. 2 *Schema cards showing query results before browsing*

Base table maintenance is provided through the 'HyperSQL Base Table' card shown in Fig. 1. It drives population of the schema base tables and provides for subsequent data manipulation through the normal Oracle 'query-by-example' HyperCard interface. A special enhancement, the 'Go To Object!' button shown on each card in Fig. 2, allows a direct transition to the HyperCard object identified by any row of a base table. The population process, triggered by the 'Populate Base Tables . . .' button, results in a physical visit to each card of the identified stack in numeric order from beginning to end. Once positioned on a card, all button, field, and background information is collected and inserted into the database. Each script is analyzed to identify all handles and to determine all direct or indirect links to other

cards. Each link found is followed to determine the identity of the target card. Each card visited is displayed in the HyperCard window and information concerning the status of the traversal is inserted into the 'message box.' Base table maintenance also includes, through the 'Delete Stack(s)' button, the capability to delete all information about specific stacks from the tables of the database, or to delete all stacks.

Data manipulation is accomplished through the 'Query Expressions' card shown in Fig. 1. We have hard-wired several generic SQL statements that are of general interest to many stack developers. The 'Execute Query' button produces a tabular display of the query output. HyperCard objects identified by various columns in the displayed rows can be selected with the cursor by the user for direct branching to the identified object. Although not yet implemented in this prototype, any update statements, e.g., 'Update Fonts', can cascade their effects both to the database and to the HyperCard objects themselves.

4. RELATIONAL HYPERTEXT QUERIES

In this section, we present numerous examples of relational queries in the hypertext domain to demonstrate the greatly increased flexibility and power such searching provides. Many of the following queries are especially useful to hypertext authors. One can determine the overall consistency of a document during its construction, and check style points that may be difficult to recognise simply by browsing. Hypertext maintainers are another class of reader that will find these types of search especially valuable.

Note that the ultimate goal of querying a hypertext is to produce, in usable (often browsable) form, a list of objects matching given conditions, not simply to build new relations. A query, then, returns a list of information about objects (cards, buttons, fields, etc.), usually including the identifiers of cards that contain the objects. What one uses this list for is unconstrained. One use for the result of a query is to keep the objects in database table form and use them in subsequent SQL queries. This form is the most general representation, and can be easily manipulated to suit the needs of HyperCard authors and readers. Another useful thing to do with the result of a query is to build a new stack from the located objects. This may entail dynamic creation of new cards to hold non-card objects (like buttons) found by the query.

Our prototype implementation accesses the base tables of the database with an Oracle software product that is essentially a HyperCard stack providing a 'query-by-example' interface to SQL. The results of a query are returned as a stack of HyperCard schema

cards (see Fig. 2) with the field values filled in. One of the buttons on this query-result display will take a reader to the actual card, and browsing can begin in the target stack from there. We are currently working on another interface that bypasses the SQL schema representation cards and creates a 'new stack' as mentioned in the previous paragraph. This stack is formed by threading together all cards identified by the query with a 'next result' button, dynamically created and temporarily added to each card. This new 'substack' can be browsed just like the original stack, since all original links are preserved and presented. The value of such an interface is that there is less of a representational dichotomy between the target stack and the results of a query on that stack.

4.1 Example query classes

Several classes of useful query are illustrated in this section. The first few are each presented in English with an accompanying equivalent SQL syntax based on the schema shown in Appendix A. After these examples, the remainder of the queries are presented in English only, simply to convey the nature of the information available through our combined hypertext/database search technique.

Example query classes with SQL syntax:

Note that 'current node' here is identified by information in the relation 'CARD' for the stack being queried.

1. *Find all nodes that point directly to the current node.*

```
SELECT  card.card_id, card.stack_id, link.target_card_id,
        link.handle_name, card.card_name
FROM    card, link
WHERE   link.target_card_id = :currentCard:
        and link.target_stack_id = :currentStack:
        and card.card_id = link.source_card_id
        and card.stack_id = link.source_stack_id
        and link.direct_flg = 't'
```

2. *Find all nodes that can be reached from the current node in one direct step.*

```
SELECT  card.card_id, card.stack_id, link.source_card_id,
        link.handle_name, card.card_name
FROM    card, link
WHERE   link.source_card_id = :currentCard:
        and link.source_stack_id = :currentStack:
```

```
        and card.card_id = link.target_card_id
        and card.stack_id = link.target_stack_id
        and link.direct_flg = 't'
```

3. *Find all nodes that can be reached from the current node in two steps, direct or indirect (via a conditional link).*

```
SELECT  distinct card.card_id, card.stack_id, A.source_card_id,
        A.handle_name, B.handle_name, card.card_name
FROM    card, link A, link B
WHERE   A.source_card_id =  :currentCard:
        and A.source_stack_id = :currentStack:
        and A.target_card_id = B.source_card_id
        and A.target_stack_id = B.source_stack_id
        and card.card_id = B.target_card_id
        and card.stack_id = B.target_stack_id
```

4. *Find all nodes reachable from the current node in an arbitrary number of direct steps.*

```
SELECT  level, source_card_id, source_stack_id, target_card_id,
        target_stack_id, handle_name
FROM    link
CONNECT BY
        Prior target_card_id = source_card_id
        and Prior target_stack_id = source_stack_id
        and direct_flg = 't'
START WITH
        source_card_id = :currentCard:
        and source_stack_id = :currentStack:
        and direct_flg = 't'
```

Note: This query is not ANSI/ISO standard SQL.[2] A generalisation of this statement for recursive operations is specified in an emerging enhanced ANSI/ISO version of SQL proposed to follow SQL2.[7]

5. *Find all cards that use 'too many' font styles (for example, more than 9) in their visible buttons and fields.*

```
SELECT  card.card_id, card.stack_id,
        count(distinct button.font_name),
        count(distinct button.display_style),
        count(distinct button.font_size),
        count(distinct field.font_name),
        count(distinct field.display_style),
        count(distinct field.font_size), card.card_name
```

```
FROM    card, button, field
WHERE   card.stack_id = :currentStack:
        and card.card_id = button.card_id
        and card.stack_id = button.stack_id
        and card.card_id = field.card_id
        and card.stack_id = field.stack_id
        and button.visible_flg = 't'
        and field.visible_flg = 't'
GROUP BY
        card.card_id, card.stack_id, card.card_name
HAVING  9 < count(distinct button.font_name)
        + count(distinct button.display_style)
        + count(distinct button.font_size)
        + count(distinct field.font_name)
        + count(distinct field.display_style)
        + count(distinct field.font_size)
```

Note: A more precise formulation of this query might use a UNION expression in the FROM clause, a feature that is currently being standardised in SQL.[7]

6. *Modify the current hypertext document to update all fields with Helvetica size 14 fonts to make them Chicago size 12.*

```
UPDATE  field
WHERE   font_name = 'Helvetica' and font_size = 14
        and stack_id = :currentStack:
SET     font_name = 'Chicago', font_size = 12
```

Note: HyperSQL document modifications like the above have not yet been enabled in our prototype hypertext query tools, but such features are straightforward extensions to the SQL UPDATE statements that would be used to make these modifications to the HyperSQL schema tables.

7. *Modify all bent arrow buttons in the current stack to show the black bent arrow icon and to show the button name 'Go Back.'*

```
UPDATE  button
WHERE   icon_id
IN      (6720, 1012, 2162, 29114, 14953, 7417)
        and stack_id = :currentStack:
SET     icon-id = 7417, show_name_flg = 't',
        button_name = 'Go Back'
```

Note: HyperSQL document modifications like the above have not yet been enabled in this hyperware development tool, but such features are straightforward extensions to the SQL UPDATE statements that would make these modifications to the HyperSQL schema tables.

8. *Find all handles for a given message in the message hierarchy of the current node.*

```
SELECT  distinct handle.card_id, handle.stack_id,
        handle.handle_name, handle.contain_object,
        handle.object_id
FROM    handle
WHERE   handle.handle_name = :givenMessage:
        and handle.stack_id = :currentStack:
        and (handle.card_id = 0 or handle.card_id = :currentCard:)
```

Note: givenMessage refers to the contents of the message box.

Other queries, without SQL syntax:

— Find all cards that contain a roadmap of a location in Europe.
— Find all cards that are 3 links away from this one *in either direction.*
— Find all cards that contain both a button labeled 'phone' and a topic field called 'funding agent'.
— Find all cards that have one button connecting to both cards A and B.
— Find any card without a secure field.
— I'm at card A . . . show all cards C that contain some single button that potentially links to both A and some other card.
— I'm at card A . . . show all cards reachable on a path from here to 'Home' that have background 'plaid'.
— Show me all cards that use background 'rootgrey' but do not contain button 'next' (i.e., that button could be placed on the background if the result is empty).
— Show me all buttons that are used on more than 90% of the cards that use background 'rootgrey'.
— Show all cards in this stack that use font *Times-Roman.*
— Identify all fonts used in this stack.
— Find the fonts that are used in fewer than 10% of the cards in this stack.
— What cards are contained in this stack? (useful if you are the maintainer of a previously-developed stack.)
— Find all cards that do not have the 'Home' button in the upper left corner.

5. CONCLUSIONS

We have implemented a powerful and flexible search mechanism for HyperCard stacks. The system automatically constructs a relational database from the information available through HyperTalk scripts and then produces browsable cards from the results of SQL queries. The system was tested by generating and querying tables for several large HyperCard stacks, including the ACM Hypertext-on-Hypertext product.[1] Subsequent examination of these stacks uncovered a number of interesting properties that were not obvious to a reader without the relational search mechanism. (The sample queries presented in the previous section were formed based on this experience).

Our prototype implementation is specific to HyperCard and SQL from Oracle, but the general techniques used are applicable to any hypertext system with any relational database. Implementation was especially easy using HyperCard because it contains a built-in programming language with an understanding of its internal stack structure. For other hypertext systems we envisage a more external implementation requiring the following steps. First, a database schema must be designed that models the salient features of a document in the hypertext system of interest. In addition to representing the physical structure of a document, other tables may be constructed (depending on the projected uses for the search system) that capture specialised information for special-purpose queries; for example, our SQL schema includes a relation that captures the x and y coordinates of each button, field, etc., to allow screen layout queries. Next, an external filter program must be written, which may have to understand the document storage format (depending on what information the target hypertext system makes readily available). This filter will scan a hypertext and gather the information needed to populate the schema tables. Finally, a backend may be required to produce a special presentation of the information obtained from queries. In many hypertext systems, the backend's actions can be carried out through mechanisms provided by the system itself — specification of where in the hypertext to begin browsing, reflecting the results of an external computation within the hypertext, etc. For the most limited hypertext systems, those for which there is no mechanism for reflecting the external environment in the state of the system, the backend may need to generate an entirely new hypertext that reflects the information produced in response to the query.

The main drawback to these general techniques is the difficulty of forming SQL (or any other relational language) queries quickly. The syntax is complex, and correctly remembering the exact field names in

a schema is taxing as well. Query-by-example interfaces provide some relief for this situation, but in general relational queries and updates on hypertext may be more likely to be used as a powerful component in an authoring environment than as a heavily used browsing facility. The correct syntax for numerous useful queries can be generated *a priori* (as done in our implementation) and a pallet of them made available to authors for quick selection as they write hypertexts. Another drawback for our specific implementation is the computed links in HyperCard. A stack is no easier to analyse than an arbitrary program, so link determination is inexact without extensive flow analysis. In other hypertext systems with less flexible link representations, completely reliable tables should actually be easier to construct.

REFERENCES

1. Hypertext on hypertext: Special issue of the June 1988 *Communications of the ACM.* Version for Apple HyperCard.
2. AMERICAN NATIONAL STANDARDS INSTITUTE. *American National Standard–Database language SQL with integrity enhancement.* Number X3.135-1989, ANSI, 1989. Revision of ANSI standard X3.135-1986.
3. APPLE COMPUTER, INC. *HyperCard User's Guide.* Apple Computer, Inc., 1987.
4. APPLE COMPUTER, INC. *HyperCard Script Language Guide: The HyperTalk Language.* Addison-Wesley, 1988.
5. ORACLE CORPORATION. *Oracle for Macintosh: References,* 1988.
6. DELISLE, N. *and* SCHWARTZ, M. Neptune: A hypertext system for CAD applications. *In: Proceedings of ACM SIGMOD '86.* Washington, DC: 1986, 132-143.
7. INTERNATIONAL STANDARDS ORGANIZATION. ISO/IEC *draft proposed database language SQL2.* Number JTC1/SC21 N3155, ANSI, January 1989. Proposed revision of ISO standard 9075:1989, the international equivalent of ANSI standard X3.135-1989.
8. MARCHIONINI, G. *and* SHNEIDERMAN, B. Finding facts vs. browsing knowledge in hypertext systems. *Computer, 21* (1), 1988, 70-80.
9. SCACCHI, W. On the power of domain-specific hypertext environments. *Journal of the American Society for Information Science, 40* (3), 1989, 183-191.
10. SHNEIDERMAN, B. User interface design for the Hyperties electronic encyclopedia. *In: Proceedings of Hypertext '87,* 1987, 199-204.
11. STOTTS, P.D. *and* FURUTA, R. Petri-net-based hypertext: Document structure with browsing semantics. *ACM Transactions on Information Systems,* 7 (1), 1989, 3-29.

APPENDIX A: SQL DATABASE SCHEMA FOR A HYPERCARD STACK

Our prototype database construction system uses an SQL schema that separately models seven important features of a HyperCard stack:

button, field, card, background, stack, link, and handle (for script representation). The individual SQL representations follow:

```
Create Table CARD (
        card_id             integer not null,
        stack_id            integer not null,
        bkgrnd_id           integer not null,
        card_name           char(40),
        card_number         integer,
        nbr_card_buttons    integer,
        nbr_bkd_buttons     integer,
        nbr_card_fields     integer,
        nbr_bkd_fields      integer,
        cantdelete_flg      char(1),
        showpict_flg        char(1),
        nbr_handles         integer,
        script_length       integer
        )

Create Table BUTTON (
        button_id           integer not null,
        card_id             integer not null,
        stack_id            integer not null,
        bkgrnd_flg          char(1),
        button_name         char(40),
        autohilite_flg      char(1),
        hilite_flg          char(1),
        showname_flg        char(1),
        icon_id             integer,
        button_style        char(15),
        button_number       integer,
        right_edge          integer,
        top_edge            integer,
        left_edge           integer,
        bottom_edge         integer,
        textalign_flg       char(1),
        font_name           char(15),
        line_height         integer,
        font_size           integer,
        display_style       char(10),
        visible_flg         char(1),
        nbr_handles         integer,
        script_length       integer
        )

Create Table FIELD (
        field_id            integer not null,
        card_id             integer not null,
        stack_id            integer not null,
        bkgrnd_flg          char(1),
        field_name          char(40),
        field_style         char(15),
        field_number        integer,
        right_edge          integer,
        top_edge            integer,
        left_edge           integer,
        bottom_edge         integer,
        autotab_flg         char(1),
        locktext_flg        char(1),
        showlines_flg       char(1),
        widemargins_flg     char(1),
        textalign_flg       char(1),
        font_name           char(15),
        font_size           integer,
        line_height         integer,
        display_style       char(10),
        scroll_pixels       integer,
        visible_flg         char(1),
        nbr_handles         integer,
        script_length       integer
        )

Create Table HANDLE (
        handle_name         char(40) not null,
        card_id             integer not null,
        stack_id            integer not null,
        contain_object      char(12),
        object_id           integer,
        handle_length       integer
        )

Create Table STACK (
        stack_id            integer not null,
        stack_name          char(40) not null,
        folder_path         char(80) not null,
        nbr_cards           integer,
        nbr_bkgrnds         integer,
        size_kb             integer,
        free_kb             integer,
        cantdelete_flg      char(1),
        cantmodify_flg      char(1),
        nbr_handles         integer,
        script_length       integer
        )

Create Table BACKGROUND (
        bkgrnd_id           integer not null,
        stack_id            integer not null,
        bkgrnd_name         char(40),
        bkgrnd_number       integer,
        nbr_buttons         integer,
        nbr_fields          integer,
        cantdelete_flg      char(1),
        showpict_flg        char(1),
        share_bkgrnd        integer,
        nbr_handles         integer,
        script_length       integer
        )

Create Table LINK (
        source_card_id      integer,
        source_stack_id     integer not null,
        target_card_id      integer,
        target_stack_id     integer not null,
        source_object       char(12),
        object_id           integer,
        handle_name         char(40),
        direct_flg          char(1)
        )
```

AUTHOR'S NOTE

This paper is a contribution of the National Institute of Standards and Technology. It is not subject to copyright. The mention of commercial products in this paper is to describe the research environment and does not imply recommendation or endorsement by the National Institute of Standards and Technology.

The portion of this work carried out at the University of Maryland was supported in part by a grant from the national Science Foundation, CCR-8810312.

The research by P. David Stotts was supported in part by the NASA Center of Excellence in Space Data and Information Systems, and the University of Maryland Institute for Advanced Computer Studies.

A HYPERMEDIA TUTORING SYSTEM FOR DESIGN TRAINERS

PHILIP J. GARTSHORE
Senior Information Technology Research Engineer,
Southern Electric, P.O. Box 13, Bartons Rd, Havant PO9 5JB

Research and development has been undertaken on the production of hypertext-based authoring and end-user training systems, linked to a videodisc player. A hypermedia system enabled the use of animation, purpose-designed graphics, sound and fast response to enhance the learning environment. The package structure could be presented, to allow the user to move around the package, and to see at a glance the consequence of his actions. In addition, material visited by the user and decisions taken, could be stored to enable computer managed manipulation of the data structure for individual requirements.

This paper discusses one particular system for design trainers to explore various structures of training material. This videodisc-based package gives guidance on the use of multi-media authoring packages to produce material that is sensitive to the interests of each trainee at an appropriate level of understanding. The control structure required some innovative programming techniques of benefit to other researchers investigating the potential of hypermedia.

INTRODUCTION

Computer-managed learning systems can modify delivery criteria, since they can: structure the learning material; schedule the management of user interaction; and store a user model to modify delivery criteria. Such developments have ranged from the simple control of video materials, to full expert systems with a high degree of interaction, responding to individual need.[1,2]

These developments are labour- and capital-intensive requiring a multi-disciplinary team, including a video producer, computer programmer, systems designer and subject specialist. The person with the greatest knowledge about content — the subject specialist — is often isolated from the technical and design processes which cause an interactive package to come into being. Subject authors and trainers are often accustomed to a sequential style of delivery (lectures, slide-tape presentations, hand-outs, and traditional video), and need to understand a system that can adapt to user requirements and can provide diverse routes to learning.

A truly interactive system cannot arise from such a rigid structure: the most appropriate packages arise from a framework in which subject authors can manipulate the manifestation of the system prior to the specification and production of multi-media materials. Some applications, however, have demonstrated that a subject author can have a satisfactory input into the final package.[3]

Hypermedia systems have brought closer the content and production sides of a training package, in that interactive material can be structured using a flexible interface with only a modest understanding of programming. The ability to control several media forming a common interface has overcome the problem that 'No single medium can tackle the range of learning environments of learning requirements and teaching approaches needed in the field of vocational training'.[4] Such an improvement does however present an overwhelming resource of facilities, requiring more guidance to be given to the subject author, whilst hiding technical aspects which are unfamiliar to the user (including the control of a videodisc or CD-ROM, sound generation, and the storage of the user model). The subject author is then free to develop training material without technical limitations.[5]

Beyond the functioning of a hypermedia system, further help is required to enable the production of the complex network of interactive modules required in a training package. User requirements can be simplified by specifying a set of characteristics representing distinct styles of learning.[6] It is now recognised that individual patterns of learning can be incorporated into subject-specific computerised learning environments.[7] Such approaches have already been used in the production of interactive videodisc packages[8], in which four parallel structures of learning had characteristic information presentation and control structures. A completely manipulatable system will require transputer-based hardware for the parallel processing of the data structure, the user interface and student modelling.[9]

Hypermedia can be controlled within distinct structures whilst allowing the user to chart an individual route. This paper describes one package — 'Training the Trainers' — in which a hypertext system was directed at trainers of design students to teach them about different structures of information best suited to individual need. An iconic interface forms a model for control structures and presentations that could enhance any interactive training system developed by a subject author.

The package introduces hypermedia by following established principles of matching '. . . not only the content of a course, but what kinds of learning' to '. . . media selection and use'.[4] The means by which it is exploited is still a matter for debate, ranging from the view that 'analytic procedures or methods are used to design, develop and evaluate instruction',[10] to a sociological stance that 'It is not possible to create a critically neutral set of [hypertext] linkages, because the act of linking involves an assertion of relationship and meaning'.[11] By

using this training system, design trainers can employ hypermedia within their own subject, undaunted by the challenge of this uncharted learning resource.

TRAINING THE TRAINERS: SPECIFICATION OF THE HYPERMEDIA SYSTEM

Previous research confirmed that trainers need direct guidance in forming strategies for the construction of a network of interactions in a multi-media training system[5,8]. Authoring systems are only useful in the sense that they provide a framework for a training package. What they lack is support to the trainer on how to produce material using the potential offered by the development environment.

Other developers of hypermedia systems employing interactive videodisc have come to the conclusion that because the general understanding about the application of hypertext is still at an early stage, there is a clear need to experiment and develop the range of strategies with which developers of multi-media systems can approach interactive material.[12] This would include the subject content and the conceptual links that are represented by hypertext linkages. It is clear from other authors that a hypertext system must have clear aims in terms of its usability before any attempt is made to construct the presentation and functioning of the final system.[13]

To explore methodologies for hypermedia development, the Design Information Research Team specified the hypertext-based training package for design trainers to learn about various ways in which design students learn, together with appropriate styles of information. This package had two aims: to explore the potential of hypermedia, and to enable design trainers to create their own learning packages.

In broad terms, this 'Training the Trainers' package required certain integral features:

(1) An iconic interface applicable to a wide range of control structures.
(2) The control of relevant material stored on interactive videodisc.
(3) A series of content and structure maps to indicate the major elements of subject content, to meet the requirement that sufficient guidance be given and that the hypertext system avoids hidden biases.[13]
(4) Direct links between maps and relevant points in the package to enable design trainers to move between items of interest, and to compare the implications of one aspect with another.
(5) Guidance on the use of the controls.

(6) Detailed information on four distinct styles of information which the design trainer can explore.
(7) To encourage design trainers to develop their own applications, using information styles and multi-media technology.

Technical specification

The 'Training the Trainers' package was implemented using HyperCard version 1.2.1 on an Apple Macintosh II with 1 Mbyte RAM and a 40 Mbyte hard disk. The videodisc player was a Philips LVROM player, controlled by an XCMD resource developed by other workers.[4] Sound resources were captured by the MacRecorder™ system. The package was initially developed for a two-screen system, for separate presentations for the videodisc image, and the iconic interface from HyperCard.

The iconic interface

Fig. 1 shows the major features of the iconic interface to the package. This interface was presented throughout the package to promote familiarity, as suggested by earlier studies.[8] Four major functions are present:

(1) User Control

These functions were available at all times through the package:

A Direction Menu containing five icons:

The Forward Arrow:	to step through the package along a main storyline, or to return to the storyline after investigating other routes through the material.
The Back Arrow:	to return to a previously visited sequence.
The Down Arrow:	to obtain more detailed information on a topic
The Context Arrow:	to obtain the purpose of the material under study
The Stop Point:	to halt the action in a motion sequence, jumping forward to the next card if in an automated hypertext sequence.

The Help Icon: to move to the set of package maps, giving guidance on the use of the package and the structure of the material.

Template Icon: to go the map of the introduction to the controls

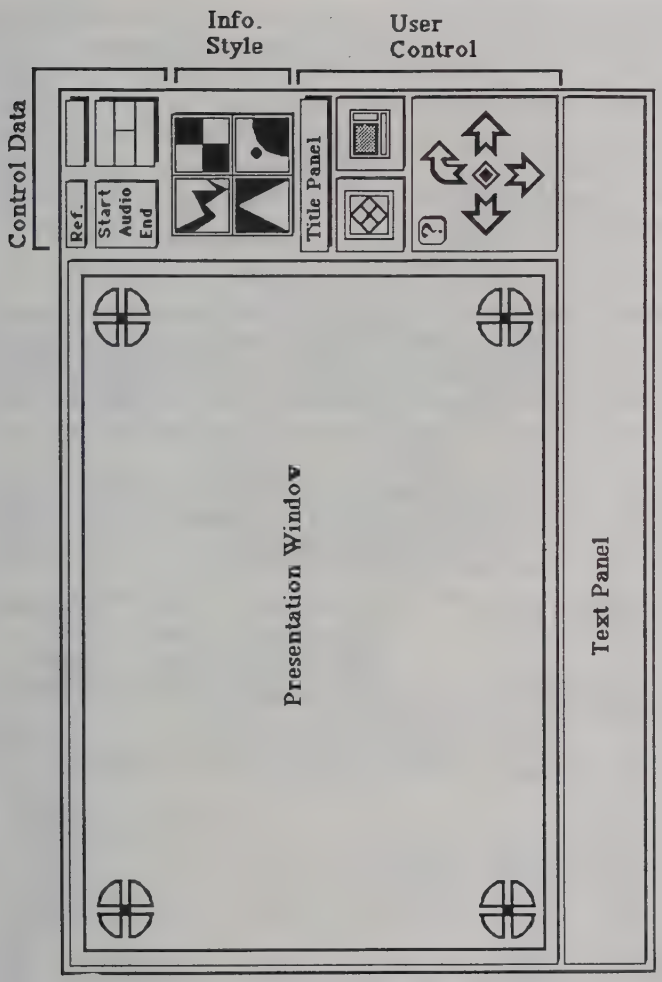


FIG. 1 *The interface to the 'Training the Trainers' package*

Menu Icon: for the overall map of the chapters of the package, enabling the user to jump from chapter to chapter.

Information Style icon: to indicate which one of four information styles is relevant to the material presented.

(2) Textual guidance

Title Panel: for confirmation of the Chapter currently studied.

Text Panel: for detail on the suggested route to be taken, and for a brief summary of the scope of the subject matter.

(3) Presentation window

Subject-specific information relating to the material presented from the videodisc, providing: maps of available routes; screen-activated menus (designated HyperCard buttons); or detailed background information.

(4) Control data

Containing references for videodisc control, including videodisc start and stop frames, audio channel and a flag for whether the video image is to be displayed or hidden. These data fields were intended both for the development team and for the design trainer to explore the inter-relationship between HyperCard control and videodisc presentation.

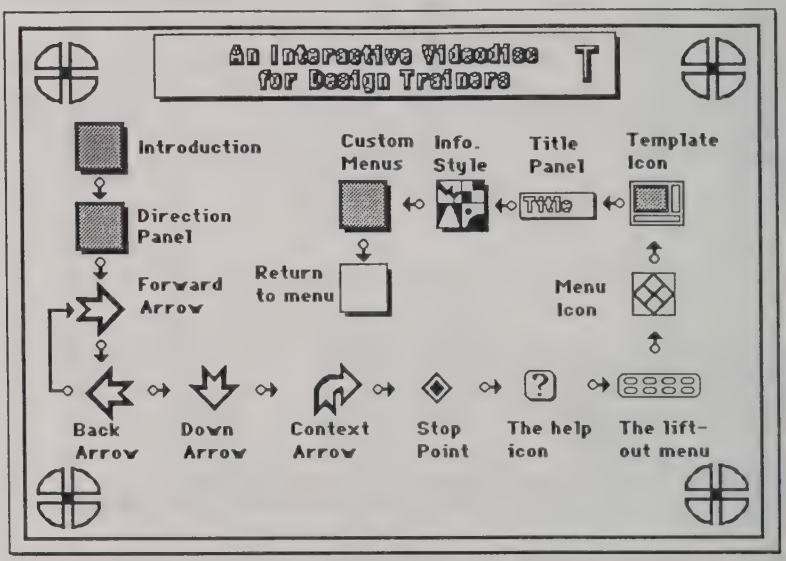


FIG. 2. *Overview of the introduction to the iconic interface*

Each icon was explained by material accessed directly from a map of the introductory sequence. Figure 2 shows the map of the iconic controls, which was available at all stages. The introduction ensured that users were familiar with the 'active' elements of the HyperCard screen, without wasting time clicking areas of the screen used only for presentation.

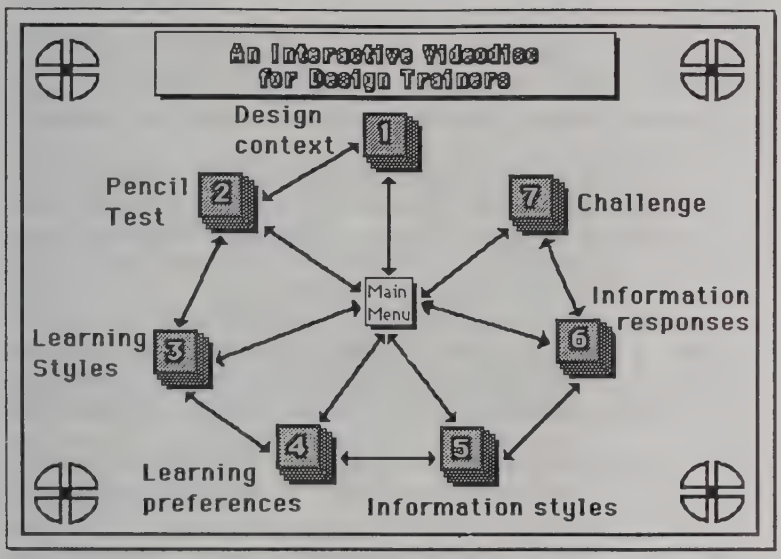


FIG. 3. *An overview of the 'Training the Trainers' package*

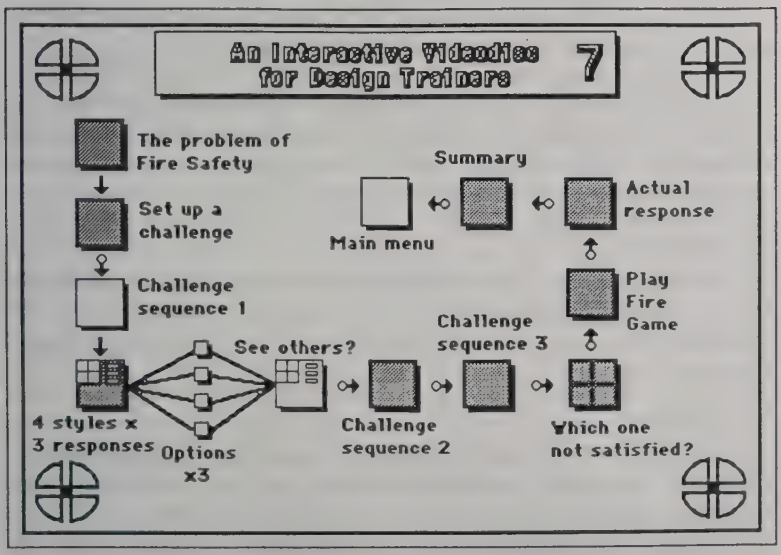


FIG. 4. *An example of a chapter map*

Overview of the control structure

The package was controlled by the user at various levels. The major routes which could be taken are represented as follows:

(1) Sequential stepping through the interactive material, chapter by chapter.
(2) A series of maps at different levels, all of which can be reached directly by the user through the action of a single icon incorporated into the interactive template. There are three types of map: a Package Map (Fig. 3) accessed by the Menu Icon, a Chapter map (Fig. 4) accessed from the Package Map or from the Help Icon, and the Map of the iconic controls (Fig. 2) using the Template Icon.
(3) Jumping to a different part of the package by going to a particular map, and activating a highlighted area. Such highlighting is considered to be of utmost importance by other developers of hypermedia systems.[13]
(4) Within Chapters a variety of control structures was made available: the nature of the interactions was indicated on the Chapter map.

SUBJECT CONTENT

The package was divided into seven chapters representing a progression of ideas on styles of learning, from initial considerations of children learning to observations of designers with particular characteristics. This sequence was not a rigid framework: through the use of Package and Chapter maps, the user could browse through the material to examine different perspectives on the issues of learning styles and information preferences; thus the trainer could re-examine a topic previously seen, then return to the current chapter. This process successfully combined a discernible structure with flexible modes of control, through a set of seven chapters:

(1) Design context

To introduce the user to the idea that people differ, and that they have their own perspectives on design problems. Several interactive structures are presented to challenge the user to explore the package in their own way.

(2) Pencil test

To emphasise the point that there are different approaches to problems. The Pencil test was presented as a tentative view that people differ in tending to use either the left or right hemispheres of the brain. The user is encouraged to try the test, then to explore their own conclusions.

(3) Learning styles

Sequences of school children in the classroom are presented to exemplify various approaches to learning. Four distinct strategies are described to link with findings by Kolb[15] that people learn by four main methods: sensing the world, watching it, thinking about it and doing something about it. Each learning strategy is represented by an activity: at the end of each sequence the user can choose one of the four methods which best matched the activity: an explanation is given of the validity of each selection made. The views of researchers and educators can be explored.

(4) Learning preferences

Interviews with some design students demonstrate how they make decisions on the basis of their own experience. The message is given that in order to operate with competence, decision-makers find it necessary to bind themselves to particular strategies.

(5) Information styles

A mime artist is used to characterise designers who operate in each of four strategies. In each case, the mime artist acts out the characteristics of each strategy in the way an architect would operate in his office. The user can then compare characteristics through a screen-activated menu structure. The mime artist exaggerated characteristics without giving offence.

(6) Information responses

A questioning framework centres on how an understanding of the four characteristic strategies is useful to the design trainer. Through the choice of strategies the design trainer can motivate students to recognise appropriate solutions to problems. A discussion between an engineer and an architect is presented to show that in the real world, people adopt mixtures of styles, which can cause conflicts. Descriptions are made available which provide details on the types of information suitable for each preferred style.

(7) A design challenge

The approaches of the four caricatures (from Chapter 5) towards information was confirmed through the use of a series of sequences relating to one particular design problem — the construction of buildings for fire safety. For each of three sequences, videodisc material is played: the user is then given the opportunity to select a caricature to perform an off-the-cuff response, a body language view, or body language with textual explanation. Fig. 5 shows an example of

the complexity of interaction available. The user is free to jump from one caricature to another, to compare responses to the material.

Finally the user is encouraged to use an interactive videodisc simulation package,[16] then to discover the caricature who was satisfied with a simulation approach, again from a study of off-the-cuff responses, body language and textual explanation. To complete the package, the four caricatures were brought back to give their views on the whole package.

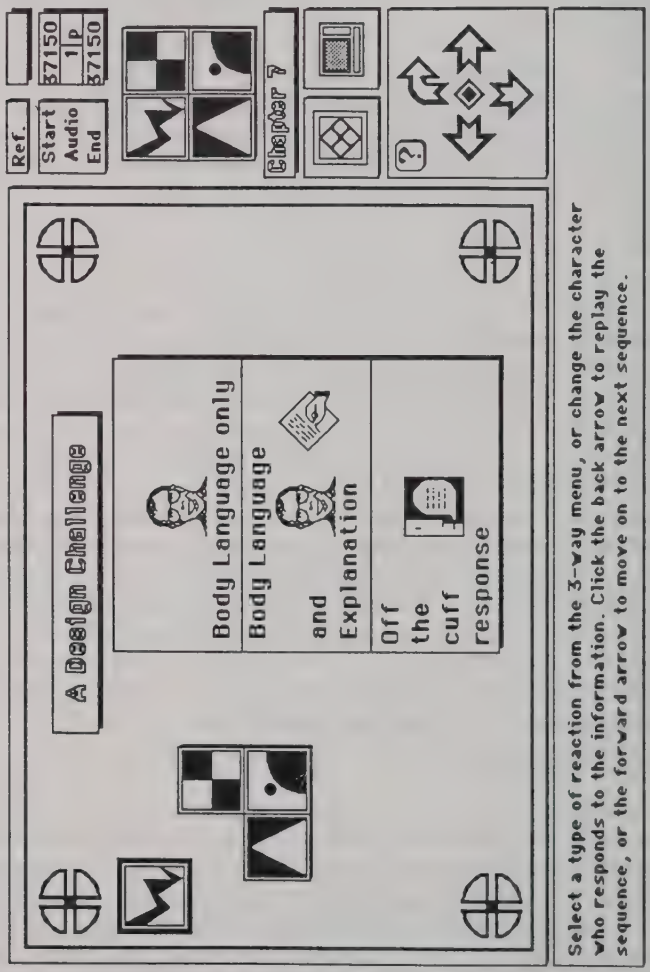

Ref.
Start
Audio
End
37150
1|p
37150
Chapter 7
?
A Design Challenge
Body Language only
Body Language and Explanation
Off the cuff response
Select a type of reaction from the 3-way menu, or change the character who responds to the information. Click the back arrow to replay the sequence, or the forward arrow to move on to the next sequence.

FIG. 5. *An example of an interactive structure from the 'Training the Trainers' package*

USER EVALUATION

It is clear that the potential of the use of hypermedia in the production of learning environments is in its infancy, and that methodologies need to be developed in formulating strategies for the manipulation of multi-media elements. An important element of this growth of ideas is the use of user evaluation to test the appropriateness of the environment created and its ease of use. By such means it will be possible to ensure that systems are not just developed for idle curiosity, or for aesthetic pleasure, but will enhance the learning opportunities for the range of potential users.

During the course of the development of the 'Training the Trainers', a pilot user evaluation programme was devised to cover a range of interests from design students, to experienced designers of interactive courseware.

Initially it was discovered by most users that their greatest difficulty was in choosing their own route through the material. In most cases users were content only to progress through the package along the suggested route. The user is encouraged to dip back in to appropriate chapters at the end of each chapter. However, in general, users did not tend to do this until they had progressed through all seven chapters.

Users claimed to have a high degree of personal involvement once they had established the purpose of the material, and had become familiar with the interactive controls.

CONCLUSIONS

The hypermedia package 'Training the trainers' on learning styles and information preferences demonstrated that hypertext can produce a versatile training package linked to multi-media devices, in this case the interactive videodisc and computer-generated audio/speech output. Hypertext enabled a sophisticated user interface enmeshed in an iconic control structure at a range of conceptual levels. Furthermore it was possible to represent the structure of the system as an integral part of the control mechanism, enabling rapid movement of the user through various parts of the package, and giving the opportunity for the user to follow their own route through the material.

The videodisc material was enhanced by the authoring functions of HyperCard. In particular it was possible to set up control functions as an integral part of appropriate cards in the HyperCard stack, to allow refinements to be made in the control data.

Visual effects were found to be effective in scheduling exciting sequences of HyperCard graphics, sound and videodisc material

(images and sound). Furthermore the automatic paging through HyperCard in association with the automatic control of the multimedia resources produced highly responsive interactions, with inherent interruptability.

The principles established in this research and development project need to be explored in further packages. Such developments would be enhanced by the introduction of CD-ROM for the storage and retrieval of video, graphics and sound, and potentially by the use of a transputer-based manipulation of the control structure, linked to a user model to manipulate the structure and content of the package in real-time through the processing of flexible database networks.[9] Such technological enhancement will meet a requirement for a 'competent [user] supervisor . . . which advises the users on how to explore the knowledge based according to their learning objectives, role, prerequisites and the learning material made available by the system'[2]. This combination of expert system and user interface management is the way forward for training systems employing hypermedia.

ACKNOWLEDGEMENTS

I acknowledge financial support from the National Advisory Board for public sector education. I am most grateful to my colleague, Jane Young, for the structure and production of the audio-visual material. I also acknowledge the invaluable contribution of Professor James Powell and Paul Newland for collaboration and discussion on the presentation of the final system. Finally I am grateful to the subjects of the User Evaluation exercise for their time and honesty. This paper is based on a presentation made at the 11th Information Retrieval Research Colloquium of the British Computer Society, 5-6 July 1989, Huddersfield Polytechnic.

REFERENCES

1. GARTSHORE, P.J., COLLACOTT, H.C. *and* POWELL, J.A. Interactive information transfer for architects. *In: Translating research into practice:* Proceedings of the CIB-86 Conference: 'Advancing Building Technology', Vol. 8. Washington DC: CIB, 1986, 3658-3665.
2. MIDORO, V., CHIOCCARIELLO, A. OLIMPO, G., PERSICO, D., SARTI, L. *and* TAVELLA, M. Interactive video and artificial intelligence: a convenient marriage. *PLET, 25 (4),* 1989, 299-309.
3. MESKELL, P. The productive use of subject matter experts in the development of CBT courseware. *Interactive Learning International, 2* (3), 1989, 8-10.
4. BATES, A.W. Teaching, media choice and cost-effectiveness of alternative delivery systems. *Vocational Training, 1,* 1988, 27-33.
5. GARTSHORE, P.J. Hypertext as a programming environment for the authoring and production of interactive videodisc training packages. *In: Proceedings of the Hypertext II Conference, University of York, 5-6 July 1989.*

6. NEWLAND, P., POWELL, J.A., CREED, C. Understanding architectural designers' selective information handling. *Design Studies, 8* (1), 1987, 2-16.
7. LECLERQ, D. *and* PIERRET, D. A computerised open learning environment to study intrapersonal variations in learning styles: DELIN. Paper presented to the Sixth Conference on Technology and Training: *Education into the 90's: Challenges of the new information technologies.* Orlando, Florida, USA, 21-23 March 1989.
8. GARTSHORE, P.J. *and* COLLACOTT, H.C. Visions of videodisc potential. *Design Studies, 8* (2), 1987, 70-75.
9. GARTSHORE, P.J. A message-passing protocol for the manipulation of multi-media databases. Paper presented at the SERC/DTI conference *Applications of Transputers,* University of Liverpool, 23-25 August 1989.
10. O'NEILL, A.F. The current status of instructional design theories in relation to today's authoring systems. *Interactive Learning International, 2* (3), 1986, 23-26.
11. DOLAND, V.M. Hypermedia as an interpretive act. *Hypermedia, 1* (1), 1989, 6-19.
12. WHITBY, M. Brave new worlds. *MacUser,* June 1989, 28-35.
13. HARDMAN, L. Evaluating the usability of the Glasgow Online Hypertext. *Hypermedia, 1* (1), 1989, 34-63.
14. CARR, L.A. *Hypercard extensions for multimedia databases.* Southampton: Department of Electronics and Computer Science, Southampton University, 1988.
15. KOLB, D. *The learning style inventory technical manual.* Boston, USA: MacBer and Company, 1976.
16. POWELL, J.A., CREED, C. *and* SIME, J.D. Escape from burning buildings: a videodisc simulation for use in research and training. *In:* J.D. Sime, *ed. Safety in the built environment,* London: E & F Spon, 1988, 87-98.

Interacting with Computers

The interdisciplinary journal of human-computer interaction

The objective of the study of human-computer interaction (HCI) is to understand how people communicate with computers so that the design and use of computer systems can be improved. HCI is an increasingly important, but diverse and fragmented field. A new medium is needed to allow the communication and co-ordination of expertise, findings and activities. *Interacting with Computers* is the international forum for just such communication. It makes information accessible not only to those engaged in research but to all HCI practitioners.

Coverage

- Systems and dialogue design
- Evaluation techniques
- User interface design, tools and methods
- Empirical evaluations
- User features and user modelling
- New research paradigms
- Design theory, process and methodology
- Organizational and societal issues
- Intelligent systems
- Training and education applications
- Emerging technologies

0953–5438 Published three times a year

Interacting with Computers is published by Butterworths on behalf of the British Computer Society Human-Computer Interaction Specialist Group.

For further details or a FREE sample copy, please write to:

Donna McMahon, Butterworth Scientific Ltd., Westbury House, Bury Street, Guildford, Surrey GU2 5BH, UK

Telephone: (0483) 300966 Fax: (0483) 301563 Telex: 859556 SCITEC G

Butterworths

USING AN ISSUE-BASED HYPERTEXT SYSTEM TO CAPTURE THE SOFTWARE LIFE-CYCLE PROCESS

MARK LEASE, MAC LIVELY, *and* JOHN LEGGETT
Department of Computer Science, Texas A&M University, College Station, Texas 77843-3112

Hypertext and object-oriented database technology offer significant possibilities for designing new software engineering environments which better represent the complex relationships between the many artifacts created during the life of a software system. These artifacts are composed of structured (machine understandable) and unstructured information. Hypertext systems can record both the structured and unstructured information of the software project and the linking relationships between both types of information. Issue-based information systems (also called argumentation systems) are special-purpose hypertext systems for capturing the issues and reasoning for decision making. The Issue-Based Experiment (IBE) system is an issue-based information system implemented on an object-oriented database platform. IBE was developed to research methods of capturing the life cycle of a software project in an issue-based information network. IBE is demonstrated to be useful in both the requirements development and system maintenance phases of the software life cycle. Several issues are discussed concerning the usability of issue-based information systems: maintaining a constant level of detail, scoping an Issue for placement in the issue network, and enforcing usage. IBE looks promising when integrated as part of a software development environment based on hypertext and object-oriented database technology.

1. INTRODUCTION

Hypertext and object-oriented database technology offer significant possibilities for designing new software engineering environments which better represent the complex relationships between the many artifacts created during the life of a software system: the customer requirements, design specifications, source-code definitions, testing information, requests for changes, and patches to the system. The information contained in some of these artifacts is highly structured and is understandable by the computer system. Source code text is an instance of structured information. However, much of the information in life-cycle artifacts is unstructured. Customer requirements and other documents in natural language text and graphics are usually unstructured. Hypertext systems allow recording both the structured and unstructured information of the software project and the linking relationships between both types of information.[1] Object-oriented databases can efficiently manage the storage of data with complex inter-relationships and provide the power of object-oriented programming.[2]

Issue-based information systems (also called argumentation systems) are special-purpose hypertext systems for capturing the issues

and reasoning for decision making in an information network. The information network explicitly represents the decisions made in the design process and the reasons for the decisions. (In this paper, the term *design process* refers to any creative activity affecting the design of the software system, including both collecting the customer requirements in the early stage of the life cycle and software maintenance in the later stages of the life cycle.)

This paper describes the Issue-Based Experiment (IBE) system, an issue-based information system implemented on an object-oriented database platform. IBE was developed to research methods of capturing the life cycle of a software project in an issue-based information network. IBE demonstrates the value of an object-oriented environment for programming and system development. By building on an object-oriented environment, we were able to implement IBE quickly (5 man-days for a working prototype) and integrate IBE into other pre-existing systems. Also, as future research finds opportunities for improvements, IBE can evolve rapidly to incorporate these modifications.

The next section describes the motivation and background for using hypertext systems in aiding software engineering. Section 3 describes the IBE system, its implementation using an object-oriented programming language and database system, and its interface to the Concordia hypertext system.[3] Section 4 looks at two examples of using IBE to capture software project information: an example of IBE recording the discussion about the design alternatives of a project using the Spiral Model[4] of software development and an example of how IBE could be extended to record the changes to a software project in the maintenance phase of the life cycle. In section 5, we present our observations on the usefulness of the IBE system and its approach. Section 6 discusses future directions of the project and further research issues.

2. BACKGROUND

2.1 Motivation for using issue-based hypertext systems in software engineering

In his description of MCC's Leonardo project, Marks reports that large, complex software systems developed with current software engineering technology are suffering a 'failure gap', a difference between the beneficial potential of the system and the actual benefits delivered by the system. The Software Technology Program at MCC is attempting to close the 'failure gap' of large, complex system development.[5]

Conklin reports many of the causes of the 'failure gap' are due to the failure to capture the design process: lost design intents, lost user intents, and problems with having a system evolve quickly enough to meet the changing demands of the users.[6] The design process generates valuable information which is lost using current software development practices. In particular, the decisions and assumptions made by the designers are extremely important, but this information is not being captured. At best, documentation is written *after* the design decisions have been made. This lost design information causes problems throughout the entire life cycle of the system.

2.2 Graphical Issue-Based Information System (gIBIS)

gIBIS is an issue-based information system for tracking *Issues, Positions,* and *Arguments*[7] that was developed by MCC's Software Technology Program. gIBIS has a graphical browser for navigating through the information network. In a typical session with the system, a user will post an Issue (for example, 'What kind of computer(s) should our department buy?'), a Position which *responds* to the Issue ('We should buy a Unix workstation.'), and an Argument which *supports* the Position ('A lot of good software is being developed for Unix workstations.'). A different user browsing the information might post another Position ('Our department should buy personal computers.') for the same Issue and an Argument which *supports* his Position and *objects to* the original Position ('Buying low-cost personal computers would allow everyone to have their own computer.').

2.3 Document Integration Facility (DIF)

The Document Integration Facility (DIF) is an Engineering Information System (EIS) for tracking the documentation produced during the life cycle of a software project.[8] DIF is part of the University of Southern California's System Factory Project, an experimental system for the development of large-scale software projects.[9] In DIF, hypertext nodes are used to store all of the information concerning the target software project. In addition to unstructured information about the target system, DIF stores the target system's requirements, and design specifications written in a machine-readable design language as well as the source code of the target system.

The granularity of a hypertext database refers to the average amount of text in the nodes of the database. Argumentation systems tend to be small-grain hypertext systems. The Document Integration

Facility is a large-grain hypertext system. Normally large-grain systems are limited in the amount of machine-understandable knowledge which can be obtained from the hypertext data because most of the knowledge is stored as unstructured information in the node. But the contents of many of the DIF nodes are written in machine-readable languages (C, NuMIL, or Gist), increasing the amount of information available to tools like NIVIZ, DIF's system configuration visualizer.

3 Issue-Based Experiment System (IBE)

The Issue-Based Experiment System (IBE) was developed to provide a platform for research and experimentation with issue-based information systems. IBE was patterned after the gIBIS system developed at MCC[7]; but like DIF, IBE will be part of a system for storing all of a project's life-cycle documentation. IBE is implemented in object-oriented Common Lisp for Symbolics workstations. Our motivation for the development of IBE is an interest in researching methods for using an issue-based information system to record all of the reasoning which is part of designing a computer software system and to provide multiple representations of this information to different stakeholders of the system.

Providing multiple representations of the information in an issue network is important. Different stakeholders of the target system will be interested in different levels of details, but the project's designers need to see these issues when they affect the parts of the project they are working on. IBE allows the user to individually select which nodes of the network are visible on the graphical display.

The ability to choose between textual and graphical representations of the issue network is also important. A graphical display of the nodes of the issue network (annotated with helpful keywords) and links (showing the relationships between the nodes) is convenient for browsing the issue network. Fig. 1 shows a sample of the IBE system displaying an issue network. Textual representations of the issue network are useful for including in hardcopy documents. Taking advantage of Symbolics' object-oriented environment, IBE interfaces with Concordia, Symbolics' development environment for technical writers.[3] Discussions about issues can be included in online and hardcopy documents through the use of Concordia's hypertext links.

Our hypothesis is that IBE will be an integral part of a software development environment designed to use object-oriented database technology. Through the object-oriented database, IBE will be able to exchange information with other system development tools such as

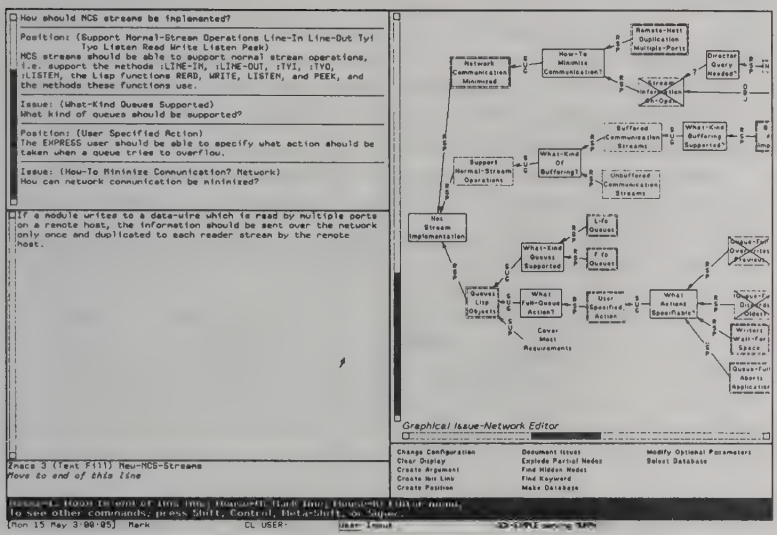

How should NCS streams be implemented?
Position: (Support Normal-Stream Operations Line-In Line-Out Tyi Tyo Listen Read Write Listen Peek)
NCS streams should be able to support normal stream operations, i.e. support the methods :LINE-IN, :LINE-OUT, :TYI, :TYO, :LISTEN, the Lisp functions READ, WRITE, LISTEN, and PEEK, and the methods these functions use.
Issue: (What-Kind Queues Supported)
What kind of queues should be supported?
Position: (User Specified Action)
The EXPRESS user should be able to specify what action should be taken when a queue tries to overflow.
Issue: (How-To Minimize Communication? Network)
How can network communication be minimized?
If a module writes to a data-wire which is read by multiple ports on a remote host, the information should be sent over the network only once and duplicated to each reader stream by the remote host.
Graphical Issue-Network Editor
Emacs 3 (Text Fill) New-NCS-Streams
Move to end of this line
Change Configuration
Clear Display
Create Argument
Create Nor Link
Create Position
Document Issues
Explode Partial Nodes
Find Hidden Nodes
Find Keyword
Make Database
Modify Optional Parameters
Select Database
(Mon 15 May 3:00:05) Mark
CL USER:

FIG. 1. *Sample screen of the IBE system editing an issue network in the Editor/Text/Graph configuration*

source-code editors, data-flow and functional-decomposition diagramming tools, and version control systems. Our ideal environment would allow anyone looking at project information in the database to access the issue-based information network which justifies the project information. The user would be able to browse and modify the issue network and traverse the issue network to see other parts of the project.

3.1 Implementation information

IBE uses Statice, Symbolics' object-oriented database system[10], to store the attribute information about Issues, Positions, Arguments, and their Links. There are no commands to save the issue-information network — incremental changes to the IBE database are made as the user executes IBE commands. This relieves the user of worries about saving the issue-information network. If either the Statice database server or the user's workstation crashes, the user will lose very little information.

All of the IBE commands are ZMACS-type commands and are bound with the hyper-super modifier. For example, the keychord

hyper-super-M makes a new database. The IBE commands are also visible on a menu pane. The normal ZMACS editor commands, including the HELP key, are available for editing text in the editor window. ZMACS is the text editor common to all system applications — IBE users do not need to learn a new editor, and their user-defined macros and commands are available while using IBE.

To impose a structure on the issue-information network, IBE limits the types of links which are legal. Fig. 2 shows which types of links are legal between the three types of nodes: Issue, Position, and Argument. The indicators for the type of link are GEN (generalizes), SPC

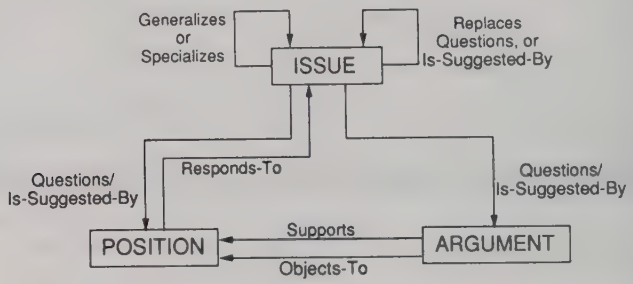


FIG. 2. *The set of legal links. Arrows indicate the direction of the link when placed in the network*[6]

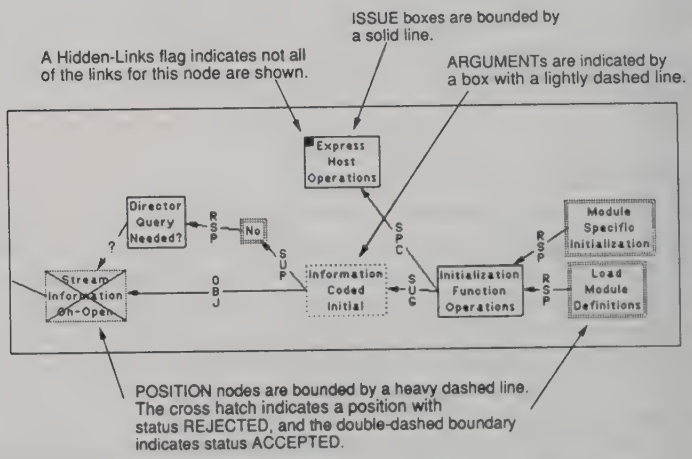


FIG. 3. *Samples of IBE nodes as they appear on the graphic-display window*

(specializes),? (questions), SUG (is suggested by), RSP (responds to), SUP (supports), OBJ (objects to), and REP (replaces).

On the graphic-display window, IBE distinguishes the type of node (Issue, Position, or Argument) by the type of line bounding the node. In addition, the status of the node is indicated graphically. The fragment of an IBE discussion in Fig. 3 illustrates the types of appearances a node may have.

3.2 Interfacing IBE to Concordia

All of the information in the issue network is available for inclusion in documents created in Concordia, and viewing or hardcopying in the Symbolics' Document Examiner. Documentation topics are created for all of the individual issues in the network. Fig. 4 shows the discussion about an IBE Issue as it would appear in Document Examiner. The documentation topic lists each Position responding to the Issue as a point under the Issue. Each Argument is listed as a point under the Position(s) the Argument is supporting or objecting to. Mouse-sensitive cross reference links are created to represent IBE links to nodes for other Issues. Clicking the mouse on the link for the 'What Algorithms Should Be used To Organize the . . .' Issue displays in Document Examiner the discussion about this Issue. When included as part of a hardcopy document, these references to other Issues are

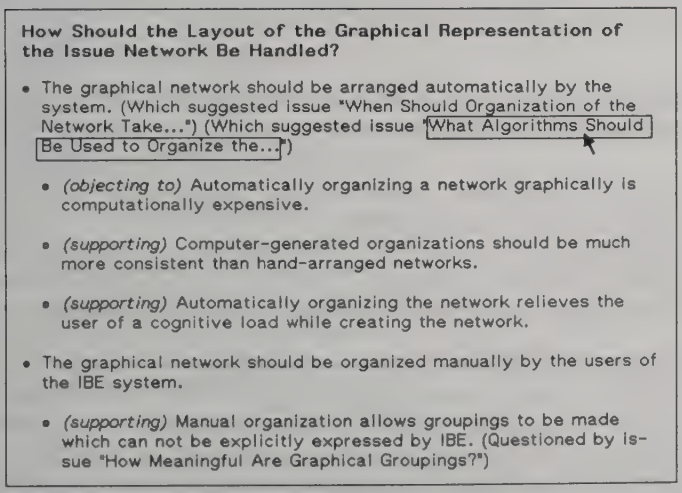

How Should the Layout of the Graphical Representation of the Issue Network Be Handled?
• The graphical network should be arranged automatically by the system. (Which suggested issue "When Should Organization of the Network Take...") (Which suggested issue "What Algorithms Should Be Used to Organize the...")
• (objecting to) Automatically organizing a network graphically is computationally expensive.
• (supporting) Computer-generated organizations should be much more consistent than hand-arranged networks.
• (supporting) Automatically organizing the network relieves the user of a cognitive load while creating the network.
• The graphical network should be organized manually by the users of the IBE system.
• (supporting) Manual organization allows groupings to be made which can not be explicitly expressed by IBE. (Questioned by issue "How Meaningful Are Graphical Groupings?")

FIG. 4. *The discussion of an IBE Issue as displayed in Document Examiner*

followed by the page number in the document where the Issue's discussion is included.

The interface between IBE and Concordia allows an IBE user to build other important views of the issue information stored in the IBE system. In Concordia, the information from IBE can be edited and rearranged to generate hardcopy and online documents targeted to different audiences. For example, the issue information related to a software system's requirements can be collected into a document browsable by the customer of the system.

When a user wants to read the discussion of an Issue in Document Examiner, the request is intercepted by IBE, which constructs the topic from the Issue's object-oriented database, and the results are displayed by Document Examiner. Interfacing IBE to Concordia was straightforward — the object-oriented environment allows IBE-generated hypertext documents to co-exist with Symbolics' online documentation without any modifications to Concordia or Document Examiner.

4. EXAMPLES OF USING IBE DURING THE SOFTWARE LIFE CYCLE

4.1 Using issue-based argumentation to implement the Spiral Model of software development

This section demonstrates the utility of issue-based argumentation systems by using IBE for discussing the objectives of a software project using the Spiral Model of software development.

Objectives	Double software productivity in five years
Constraints	$10,000 per person investment Within context of TRW culture Government contracts, high tech., people oriented, security Preference for TRW products
Alternatives	Office: Private/modular . . . Communication: LAN/star/concentrators Terminals: Private/shared;smart/dumb Tools: SREM/PSL-PSA/. . .;PDL/SADT/. . . CPU:IBM/DEC/CDC/. . .
Risks	May miss high-leverage options TRW LAN price/performance Workstation cost TRW LAN benchmarking
Risk resolution	Extensive external surveys, visits Workstation price projections
Risk resolution results	Operations concept: Private offices, TRW LAN, personal terminals, VAX

	Begin with primarily dumb terminals; experiment with smart workstations Defer operating system, tools selection
Plan for next phase	Partition effort into software development environment (SDE), facilities, management Develop first-cut, prototype SDE Design-to-cost; 15-person team for one year Plan for external usage
Commitment	Develop prototype SDE Commit an upcoming project to use SDE Commit the SDE to support the project Form representation steering group

FIG. 5. *Spiral Model usage: TRW Software Productivity System, Round 1 (from* [4]*).*

Boehm's Spiral Model of software development is a risk-driven, iterative approach to software development.[4] Each iteration of the process, a loop around the spiral, attempts to increase the likelihood of the project succeeding by reducing the risks facing the project. An iteration of the process goes through several phases:

— Determine the objectives of this phase of the project, identify the alternative solutions to achieve these objectives, and the constraints the solution must satisfy.
— Evaluate the alternative solutions and identify and resolve the risks associated with each solution.
— Carry out the solutions to develop and verify the next-level product.
— Plan for the next phase of the project.
— Each loop of the spiral is terminated with a review of the progress of the project, and a commitment of resources to carry out the next phase of the project.

Boehm and TRW applied this model to a project for developing a software productivity system which would allow their company to double their software productivity in 5 years. Fig. 5 is an activity summary for one of the initial spiral loops of the project. At this stage, the project was looking at the alternative technologies which could be used to improve productivity.

To research the usefulness of the issue-based paradigm of knowledge representation and argumentation, we used IBE to represent part of the table as an issue network. The alternative technologies are issues for discussion. The choices within the technologies are Positions responding to the Issues. The Positions have reasons for selecting or rejecting them which become supporting or opposing Arguments. The risks associated with the technologies are

themselves issues to be discussed. The network can be extended until all of the important risks have been identified and evaluated.

The process of identifying the alternative solutions and identifying the risks involved with each solution fits IBE's framework very cleanly. IBE's format of the discussion made the choices and their supporting or objecting Arguments clear.

The rest of this section is part of the text of the demonstration issue network. Except for the illustrations, the rest of this section was created by IBE from the text input into the issue network. This document was authored in Concordia.[11] *IBE generated the formatting commands for printing the information, and the IBE-generated document was included in this document through hypertext include-text links. Fig. 6 is a section of a screen image from part of the IBE issue network that generates this discussion.*

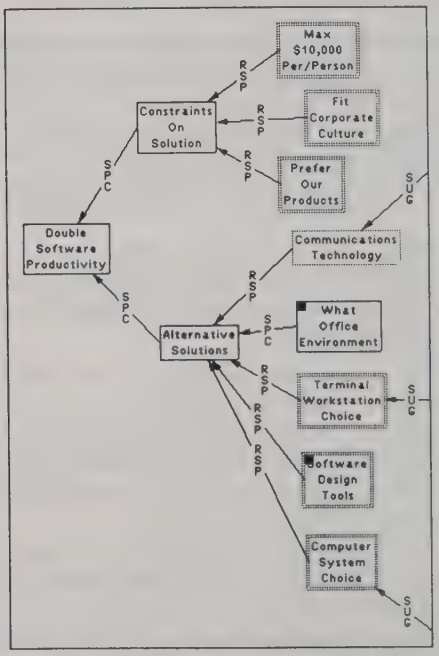


FIG. 6. Discussion of alternative solutions for achieving the goal: doubling software productivity in 5 years. The constraints of the problem are also shown

4.1.1 How should we plan to double our software productivity in the next 5 years?

What are our alternative solutions?

— Our choice of software design and analysis tools will greatly impact our productivity. (Which suggested issue 'What Productivity Tools Should Be Supported?')
— We should consider the impact of our choice of computer system on our software productivity. (Which suggested issue 'What Sort of Computer System Should the Software . . .')
— We should consider the effects of the choice of terminal/workstation on productivity. (Which suggested issue 'What Kind of Computer Terminals Should Be Used to . . .')
— The kind of communications technology we use will effect our productivity. (Which suggested issue 'What Kind of Communication Technology Should Be . . .')

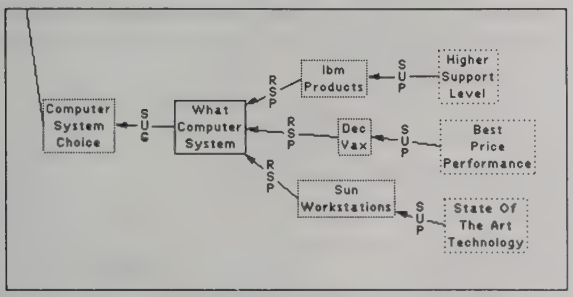


FIG. 7. *Discussion of alternative computer systems and their benefits*

For our problem of doubling software productivity, what constraints must our solution exist within?

— The solution must fit the existing corporate culture: government contracts, high technology, people oriented, security-sensitive projects.
— The solution must show a preference for our own products.
— The investment per person should not exceed $10,000.

4.1.2 What kind of communication technology should be used to communicate between host computers?

Host systems should communicate over local area networks.

— *(supporting)* Local area network protocols are widely standardised among system manufacturers.

— *(supporting)* LANs can be hooked up to wide-area networks for transparent inter-network communication.
— *(objecting to)* LANs are easily tapped, and detection is difficult. (Questioned by issue 'What about encrypting network packets to foil . . .')
— *(supporting)* The same communication medium can be used for host-to-host and terminal-to-host communications. (Supporting Position 'Local area networks and terminal bridges should be . . .'; responding to issue 'What kind of communication technology should be . . .')

High-speed bus to bus communications should be used.

— *(objecting to)* Bus to bus connections are very expensive.
— *(objecting to)* The maximum distance between computers is unacceptably small.
— *(supporting)* Bus to bus communications are difficult to decode, and therefore are more secure.
— *(objecting to)* Bus to bus communications are very host dependent, and would not support a heterogeneous system population.

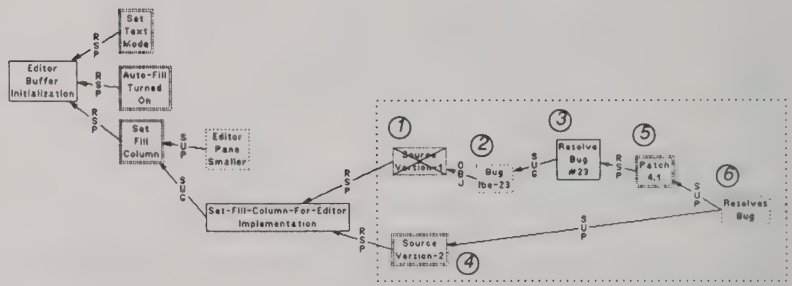


FIG. 8. *An example of using an IBE network to maintain system change-history information. The nodes are numbered in their order of creation.*

4.2 Using Issue-based Argumentation to represent a program's change history

Part of capturing the design of a system is recording the changes made over time to the system and the reasoning behind the changes. Even when the changes are non-controversial and do not require discussion, an issue-based information network provides a good medium for recording the evolution of the system. To demonstrate how this information might be expressed in an issue network, IBE was used to

simulate part of the design of a system (IBE itself) and demonstrate how a change in response to a bug report might be recorded in the issue network. Fig. 8 shows the issue network created for this example, and Fig. 9 shows the contents of the nodes in the outlined area of Fig. 8.

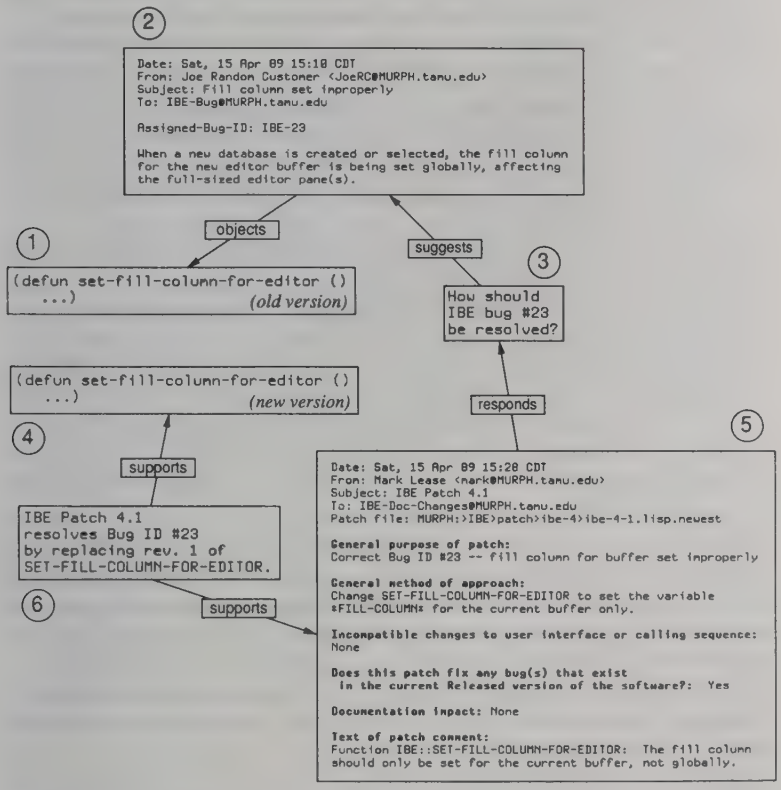

2
Date: Sat, 15 Apr 89 15:10 CDT
From: Joe Random Customer <JoeRC@MURPH.tamu.edu>
Subject: Fill column set improperly
To: IBE-Bug@MURPH.tamu.edu
Assigned-Bug-ID: IBE-23
When a new database is created or selected, the fill column for the new editor buffer is being set globally, affecting the full-sized editor pane(s).
objects
suggests
1
(defun set-fill-column-for-editor ()
...)
(old version)
3
How should IBE bug #23 be resolved?
(defun set-fill-column-for-editor ()
...)
(new version)
responds
4
5
supports
IBE Patch 4.1
resolves Bug ID #23
by replacing rev. 1 of
SET-FILL-COLUMN-FOR-EDITOR.
Date: Sat, 15 Apr 89 15:20 CDT
From: Mark Lease <mark@MURPH.tamu.edu>
Subject: IBE Patch 4.1
To: IBE-Doc-Changes@MURPH.tamu.edu
Patch file: MURPH:>IBE>patch>ibe-4>ibe-4-1.lisp.newest
General purpose of patch:
Correct Bug ID #23 -- fill column for buffer set improperly
General method of approach:
Change SET-FILL-COLUMN-FOR-EDITOR to set the variable
FILL-COLUMN for the current buffer only.
Incompatible changes to user interface or calling sequence:
None
Does this patch fix any bug(s) that exist
in the current Released version of the software?: Yes
Documentation impact: None
Text of patch comment:
Function IBE::SET-FILL-COLUMN-FOR-EDITOR: The fill column should only be set for the current buffer, not globally.
6
supports

FIG. 9. *The contents of the nodes tracking the change history in Fig. 8. The nodes are numbered in their order of creation*

The initial issue network expresses part of the low-level design for initializing IBE's editor buffer when a new database is selected by an IBE user. As part of the process of modifying the system when an error is discovered in a function, the network goes through a number of changes:

1. Initially the issue network has the source code for SET-FILL-COLUMN-FOR-EDITOR (a function for adjusting the word-

wrap column for the editor in the IBE window pane) as a Position node responding to the Issue of implementing the function.

2. A customer reports a bug caused by an error in SET-FILL-COLUMN-FOR-EDITOR. When the source of the error is located, the report is added to the network as an Argument node objecting to the current implementation of the function.

3. The Issue is raised questioning how the problem should be corrected. Possible responding Positions include redesigning higher-level functions or changing the functionality of the system. Even though the bug is caused by this function, the solution to the problem might require making changes to other parts of the system. In this case, the problem is corrected by changing this function.

4. The revised source code for the function is added to the network as a new Position node.

5. A patch file is created collecting all of the changed definitions for the software system. This patch file can be loaded by customers to correct the problem in their copy of the software system. A Position node documenting the change is added to the network responding to the Issue of correcting the problem.

6. An Argument supporting both the function's new source code and the patch documentation links the changed code and the patch together.

In a software engineering environment implementing an integrated issue-based information system, the changes to the issue network would be created automatically by the functions which performed the bug tracking and configuration management of the system. The issue network provides the medium for recording the information — information which could be used by other people inquiring on the status of the bug report, the reasons for the patch, or the reasons for the incremental change of the function.

5. OBSERVATIONS FROM USING IBE

In using IBE as a stand-alone system, we have found that designing software by generating an issue-based information network is a challenge. On the one hand, creating nodes for design issues as they are thought of is very useful. It allows an IBE user to raise issues he/she knows will need to be addressed but still keep their main chain of thought intact. However, from our experience using IBE, we have observed 4 areas of concern which must be addressed before issue-based information systems will be widely used:

— Users want to browse the issue network maintaining a constant level of detail with respect to the design.
— Determining the proper scope of an Issue for placing the Issue's node in the network can be difficult.
— To maintain a current, accurate database, the argumentation system must have a mechanism for enforcing its usage.
— Considerations about graphically arranging the issue-based network can distract from the design process.

Constant level of detail — Browsing an issue network at a constant level of detail is difficult. When system design is performed with a top-down design technique, successive descriptions of the system are created, and the descriptions are increasingly detailed. When the system's description is represented in an issue network, determining the level of detail a particular node is referring to can be difficult. Unlike functional decomposition charts which get more specific the further from the top of the chart you travel, issue networks are not easily arranged according to specificity. Some of the issues linked to a node may deal with very specific technical details while other nodes may be very general. Further research needs to be done in automatically filtering the displayed network.

Scope of an Issue — As the designer tries to specify greater levels of detail for the target system, many of the Issue nodes created will be applicable only to the particular design implementation to which the Issue is attached; but issues might be raised during the argumentation of the implementation which are more generally applicable. If the design implementation is rejected or made obsolete, the issues raised by that implementation need to be reviewed for applicability. If the issue affects more than that particular implementation, the Issue should be retained and remain active. The issues applicable only to a particular implementation will be retained for historical purposes when the implementation is rejected or replaced. Therefore, major restructuring of the target system's implementation means major restructuring of the issue network. Even if an implementation is scraped and restarted from the beginning, the user can not assume all of the old nodes are useless. Many of the Issue nodes are still relevant. The Position and Argument nodes for these retained issues need to be considered individually to see if they are still applicable to the new implementation.

Usage enforcement — A stand-alone system does not prevent the tendency to design part of the system and afterwards create the issue network for the design. This behaviour has the problem of conventional, after-the-fact documentation — the documentation tends to be incomplete and out of date. This is a hard problem to solve.

Until the specification tools become totally integrated with the software development environment, this problem will be true of all such tools.

Graphical representation — The graphical representation of the network has its own problems which need further research:

— Users of the system can create semantically equivalent issues in different parts of the network and not be aware that they have done so.
— Rearranging nodes on the graphical editor to make room for debating an Issue takes time and detracts from the thought process.
— Long distance links across the graph can be awkward.

6. FUTURE DIRECTIONS

We intend to continue to use IBE to design software systems and to research the interfacing of IBE with other types of software engineering tools. This research should start to answer the following questions about the usability of issue-based information systems:

Human factors design issues

— When multiple people are contributing to the same network, how much coordination is needed to prevent collisions while updating information?
— How much does manipulating the graphical representation detract from the main goal of representing the argument?
— Are participants and non-participants able to look at the issue network and follow the discussion?

Knowledge representation issues

— How closely does the hierarchy of top-down, increasingly-detailed design correspond to the hierarchy of the issue network?
— Can an issue-based discussion be refined into the level of detail required by the implementors of the system?
— What kind of relationships are difficult to represent in IBE format?
— Does the lack of a decision-support mechanism detract, as Conklin suggests[6], from the tool's usefulness?
— How much of a system's design can be captured in an issue network?
— Can the information in the issue network be translated into other representations, such as data-flow or state diagrams, to provide other views of the information?

- Does the object-oriented paradigm of design and programming gain or lose power in this kind of environment?
- To track changes in the issue network after the system has been specified/implemented, the out of compliance parts of the implemented system will need to be located. Also, system maintainers suspecting an error in a coded function, will require a method of tracing the function back to the parts of the issue network which give the details of the implementation of the function. How can this two-way linkage be maintained?
- What techniques can be used to represent higher-level relationships between objects? Can these relationships be expressed as patterns of IBE sub-networks or are additional node and link types needed?

Our current hypothesis is that an issue-based argumentation system will be most useful not as a separate tool but as a capability woven through the entire software engineering environment. In a software engineering environment providing hypertext-like linkages of information and an object-oriented database, a developer should be able to reference an artifact in the system (a specification, data-flow diagram, source code for a function, etc.) and access a history of the artifact's development. The history should provide the developer with the information the developer needs to understand the thinking which went into the artifact. An issue-based information system should be a useful method of building and representing the artifact's development history. Object-oriented database and hypertext technologies provide powerful tools for developing systems, such as IBE, to improve software engineering.

AUTHOR'S NOTE

All correspondence regarding this paper should be sent to John Leggett at the address given.

REFERENCES

1. GARG, P. *and* SCACCHI, W. *A software hypertext environment for configured software descriptions.* University of Southern California, Los Angeles, CA: Computer Sciences Department, August, 1987.
2. LUI, L.C. *and* HOROWITZ, E. Object database support for a software project management environment. *ACM SIGSOFT '88: 3rd Symposium on Software Development Environments, February, 1989.*
3. WALKER, J. Supporting document development with Concordia. *IEEE Computer,* January 1988, 48-59.

4. BOEHM, B.W. A Spiral Model of software development and enhancement. *IEEE Computer,* May 1988, 61-72.
5. MARKS, P. *What is Leonardo?* MCC Technical Report Number STP-141-86. Austin, Texas: Microelectronics and Computer Technology Corp, April, 1986.
6. CONKLIN, J. *Design rationale and maintainability.* MCC Technical Report Number STP-249-88. Austin, Texas: Microelectronics and Computer Technology Corp, June 1988.
7. CONKLIN, J. *and* BEGEMAN, M. *gIBIS: A tool for all reasons.* MCC Technical Report Number STP-252-88. Austin, Texas: Microelectronics and Computer Technology Corp, July 1986.
8. GARG, P. *and* SCACCHI, W. *A hypertext system to manage software life cycle documents,* Proceedings of the 21st Hawaii International Conference on Systems, Honolulu, Hawaii.
9. SCACCHI, W. The USC system factory project. *Software Engineering Notes,* January, 1989, 61-82.
10. SYMBOLICS, INC. *Statice.* Cambridge, Mass., February 1988.
11. SYMBOLICS, INC. *Concordia,* Cambridge, Mass., February 1988.

LINKING ACTIVE ANCHORS: A STAGE IN THE EVOLUTION OF HYPERMEDIA

MURUGAPPAN PALANIAPPAN, NICOLE YANKELOVICH *and* MARK SAWTELLE

Institute for Research in Information and Scholarship (IRIS)
Box 1946, Brown University, Providence, Rhode Island 02912

With the advent of multimedia technology, hypertext has evolved into hypermedia, in which the elements that are linked together can include animations, video clips, music recordings, and database queries. The critical hypertext function of link traversal must undergo a parallel evolution to support the dynamic aspects of these new types of information.

This paper introduces the concept of *active anchors* within dynamic information and explores the design issues involved in linking to active anchors. Scenarios from our hypermedia system, Intermedia, illustrate the use of active anchors in a university setting. General conclusions recommend characteristics that may facilitate the implementation of this concept in other hypermedia systems.

1. INTRODUCTION

With the advent of multimedia technology, hyper*text* has evolved into hyper*media*, extending its scope to include media other than computer-based text. Some of these media are *static,* while others are *dynamic.* Information in static media is stationary, whereas information in dynamic media changes over a period of time. Examples of static media are text, graphics, and timelines; examples of dynamic media are animations, videodisc recordings, and retrievals from online databases. We refer to applications based on static media as *static applications* and applications based on dynamic media as *action-oriented applications.*

Hypermedia systems allow authors to associate information by linking it together so that users can browse related information quickly by traversing the links. Links are created between a pair of end-points, which may be entire documents or portions of documents. The link traversal convention in hypermedia systems allows users to navigate from the source end-point to the destination end-point of a link. While this convention is sufficient for static applications, it needs to be extended for action-oriented applications. Users should be able to acquire dynamic information easily, to explore the information further, and to author links to the information in a consistent manner within the hypermedia system.

In a hypermedia environment, it is possible for many static and action-oriented applications to coexist. Since the addition of links to a multimedia environment encourages rapid transitions from one

application to another, users should not be forced to know the details of operating each application they encounter. Upon following a link, users should be presented with the information at the other end, rather than with another interface to negotiate. In other words, following a link to an animation document should not just bring up the document but should also play the animation. Users wanting to explore the information further should be provided with tools to do so. For example, users may want to view part of an animation at a slow pace. Users should be encouraged to create links to dynamic information; they should be able to do so with the same ease with which they can create links to static information.

This paper aims to:

1. Define our meaning of the concept of 'active anchor', and differentiate it from other mechanisms that may appear similar.
2. Identify some major considerations for the implementation of active anchors in action-oriented applications.
3. Illustrate the use of active anchors with examples from some action-oriented applications in Intermedia.
4. Recommend characteristics that may facilitate the implementation of linking to active anchors in hypermedia systems.

We define an *anchor* as the endpoint of a link; therefore, a *static anchor* is an insertion point, a co-ordinate location, a collection of objects, or an entire node in a static application. An *active anchor* embodies dynamic information in an action-oriented application. Link traversal to a static anchor results in navigation; link traversal to an active anchor results in navigation and the unfolding of dynamic information associated with that anchor. Examples include the playing of an animation or videodisc sequence, the recording of an audio or music sequence, or the execution of a database query.

Several existing systems allow users to specify actions, but, to do so, users must write scripts and attach them to the source end of a link. For example, in KMS, the specification can be accomplished with the KMS Action Language[1]; in NoteCards[2], a program can be written on a script card; in HyperCard[3], a script can be written and attached to some link activator such as a button. Goodman[4] provides examples of scripts for HyperCard to activate and interact with sound documents.

While scripting approaches offer rich functionality, they do not help the average user to create links to active anchors. Scripting approaches require some level of programming skill and exist largely outside the desktop user interface. Instead of allowing an application to provide

tools for the author to define the scope of an active anchor, the scripting mechanism forces the author to know the operational details of an application to define the scope.

Further, scripts are attached to the source end of a link, whereas the dynamic information is at the destination end. Altering the information at the destination end may require updating the script at each source end of a link.

The Intermedia-based active-anchor model described in this paper avoids the shortcomings of the scripting approaches by providing tools to allow an author to define an active anchor without programming. Our model also provides system-level support needed for the effective participation of active anchors in hypermedia.

2. OVERVIEW OF INTERMEDIA

Although Intermedia[5] shares many similarities in functionality with other hypermedia systems[6], it differs in at least three ways. First, the hypermedia functionality is integrated into each application so that the actions of creating and traversing links can be interspersed with the actions of creating and editing documents. Second, links are established between arbitrary selections in documents, called anchors (also referred to as blocks). This is in contrast to many other hypermedia systems where links are established between an anchor and an entire document or node. Link marker icons placed near the source and destination anchors indicate the existence of a link in Intermedia. Third, links are bi-directional. Users can traverse a link from either endpoint to the other.

The user interface for creating links within Intermedia is modelled after the Macintosh/Smalltalk Copy/Paste paradigm.[7,8] To create a link in Intermedia, the user makes a selection within a document, issues the 'Start Link' menu command, performs any number of operations before making a second selection in any document on the desktop, then finishes the task with the 'Complete Link' command.

Intermedia allows textual attributes to be attached to links and anchors to provide users with meaningful descriptions. These descriptions can be edited by the author of the document in which the anchor resides and by users with appropriate permissions. Users can edit anchor or link descriptions through property sheets. Any modifications made are persistent for all users accessing that link or anchor.

3. ACTIVE ANCHOR DESIGN ISSUES

3.1 The Need for Active Anchors

We discovered the shortcomings of static anchors while attempting to incorporate linking functionality into several of the action-oriented applications being studied and developed at IRIS:

InterVideo	a videodisc sequence editor	(implemented)
InterPlay	an animation sequence editor	(implemented)
InterVoice	a voice-sampling editor	(under consideration)
InterAudio	a CD-audio sound-bite editor	(under consideration)
InterBrowse	a database query editor	(designed)

When users traverse a link to a static anchor, Intermedia opens the destination document, scrolls it so that the destination anchor is visible, and highlights the contents of the destination anchor. When we added an action-oriented application, this functionality proved to be inadequate. Before extending the convention to include support for active anchors, we studied a number of commercial, action-oriented applications. In each case, we asked ourselves, 'if this application ran in the Intermedia environment, what would it mean to follow a link to data created with the application?' We classified applications into three paradigms.

Applications following the *playback* paradigm convey information by rendering a dynamic view of a stored set of sequential data items.

Such data items include:

— frames on a videodisc
  Examples: HyperCard (ABC News Interactive Suite, Voyager Video), TimeFrame (Neo Tech)
— tracks on a compact audio disc
  Example: CD-Remote (Apple Desk Accessory)
— frames of graphic animation
  Example: VideoWorks II (MacroMind)
— and 'bites' of sound
  Examples: MacRecorder (Farallon), Performer (Mark of the Unicorn)

Applications following the *recording* paradigm convey information that is made up of stored data items or streams fed into the application from some external sensing or recording device.

Such data items include:

— video signals
  Example: TimeFrame (NeoTech)
— audio/music sound bites
  Examples: MacRecorder (Farallon), Performer (Mark of the Unicorn)

— analog/digital sampling
  Example: LabView (National Instruments)

Applications following the *query* paradigm present information from a large collection that meet a user-defined set of criteria. In this paradigm, it is not so much the information itself which is dynamic, but the retrieval of the information.

Such data items include:

— bibliographic database queries
  Examples: BRS (BRS Information Technologies), Knowledge Finder (Aries Systems)
— full text database queries
  Examples: BRS (BRS Information Technologies), Lexus (Mead Data Central), Blue Fish (Lotus)
— online transaction systems
  Example: Sabre (American Airlines reservation system)

The action-oriented applications we have designed or considered for Intermedia fall into the playback category (InterPlay, InterVideo, InterVoice, and InterAudio) and the query category (InterBrowse). Based on both our analysis of our own applications and those designed by others, we identified a number of design issues that had to be addressed before action-oriented applications could be well integrated into the Intermedia environment. The following case study illustrates each design issue.

3.2 Case study: InterPlay

InterPlay is an animation editor that allows users to create and control two-dimensional animations by specifying a series of *marks* which are analogous to 'key frames' in movie animation. Animation characters are created with structured graphics tools at the frame at which they enter the scene. This frame automatically becomes the first mark for a character. To move a character from a position in one frame to a different position in another frame, users advance to the later frame, move the character and specify it as marked. Based on these marks, the application generates the pictures that appear at all intermediate frames. InterPlay provides the user with VCR-like controls to play back an animation. These controls include: Play, Stop, Rewind, Wrap, Step, Speed, and Frame Search.

Scenario from Intermedia

A hypothetical Russian history course is being taught with the partial assistance of an experimental version of IRIS Intermedia. To illustrate the path of Napoleon's invasion and retreat, Prof. Krzysztof

Krzyzanowski and his graduate teaching assistant, Sandra Sonderblatt, have constructed an InterPlay animation sequence.

Prof. K and Ms. S scan a bitmap image of western Russia into Intermedia's bitmap image viewer; the map is then copied and pasted into InterPlay. They trace the route of Napoleon's army, as shown in Fig. 1, using structured graphics tools selected from InterPlay's palettes.

A Grand Army symbol represents the army at the outset of its march, in Poland. As the animation progresses, the army moves east, toward Moscow. Prof. K and Ms. S create this effect by moving the symbol to various marks on the map (outside Smolensk, for instance), then choosing InterPlay's Linear *menu command to have the application move the symbol in a straight line between its current and previous locations. K and S create different symbols using InterPlay tools to emphasize the dwindling size of Napoleon's forces as his disastrous campaign progressed. When the animation sequence is finished, users are able to use the control panel to run the animation, causing the symbol to move around the map following the path defined by K and S. Fig. 2 shows an animation frame in which the army is moving from Borodino to Moscow.*

After they have finished animating Napoleon's movements, Prof. K and Ms. S begin to link their animation document to an Intermedia timeline document, shown in Fig. 3, that includes entries for the important dates in Napoleon's campaign. They construct links between dates on the timeline and the corresponding locations of the army on the map. They want students exploring the links created between the documents to be able to select one of the dates on the timeline and traverse to the animated map.

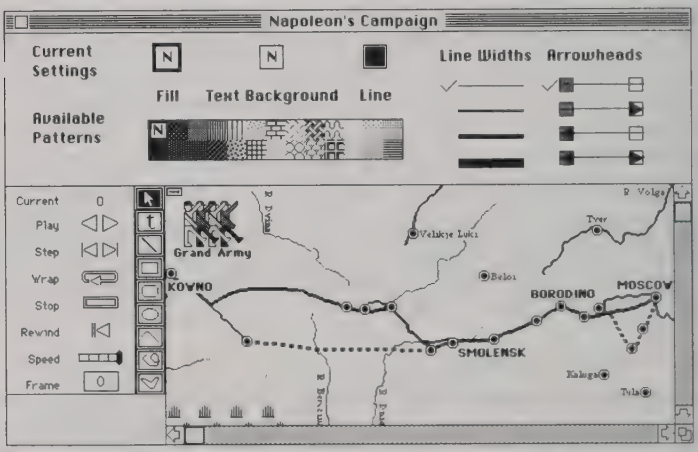


FIG. 1. *An InterPlay document showing a bitmap graphics map of the Russian Empire*

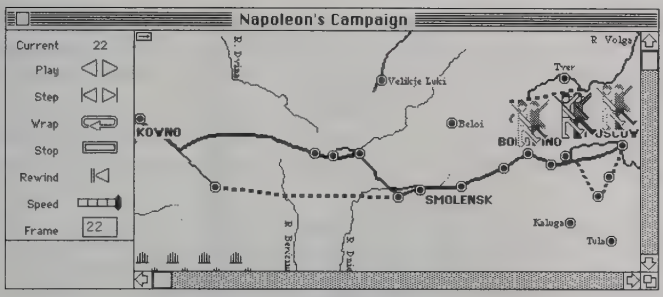


FIG. 2. *The same InterPlay document as in Fig. 1 (with the editing palettes hidden) after the animation has been partially played. The shaded line tracing the symbol's path is an author's tool that is normally hidden*

This leads us to the first important design issue.

— When users traverse a link to an active anchor, what should happen when the destination document opens?

The answer to the above question depends on the information that an author wants to convey to users and on the type of application containing the destination anchor. In our example, it is likely that the authors would want the link from the title of the timeline document to the animation to just open the animation document, displaying the first frame. On the other hand, the authors probably intended users following a link from the 'Crossing of Berezina' entry in the timeline to see the segment of the animation that illustrates the crossing.

Since playback, recording, and query documents have substantially different interfaces from one another, we have tackled this design issue by breaking the problem down into two parts. First, we have added generic functionality to the Intermedia application framework that will help developers integrate any of the three types of applications. Second, we have left it up to each application developer to decide how best to fit the specific application into our generic framework.

In terms of the generic extension to the Intermedia framework, we have defined two basic options for what can happen when a user traverses a link to an active anchor. In the first option, the destination document can open without executing any action, as is the case with static applications. It would then be up to the user to interact with the application to initiate any actions.

> Sue Cox and Bob Lee, two of Prof. Krzyzanowski's history students, are using the materials developed on IRIS Intermedia to help them understand Napoleon's invasion and retreat. Sue opens the timeline and scans the events that

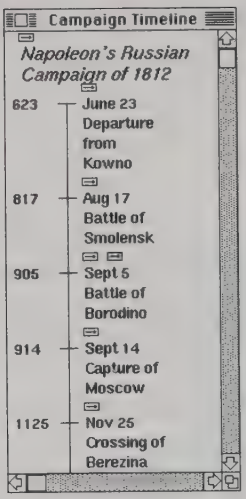


FIG. 3. *An InterVal timeline document containing a chronology of Napoleon's Russian Campaign. The arrow icons above each event indicate a link to one or more other anchors*

took place. She then traverses the link from the title 'Napoleon's Russian Campaign of 1812.' This traversal opens the animated map at the first frame. She clicks on the 'Play' button in the control panel and watches the animation until Napoleon's army reaches Moscow, at which time she decides she wants to explore other information. She halts the action using the 'Stop' button and is now able to explore other links emanating from the animation document.

The second option is for the destination document to open and *execute an action,* using the location of the link anchor as the starting point (if applicable). Deciding whether or not a link should trigger an action is up to the author of the link. For this purpose, we have added a toggle control to the Link Property Sheet authors normally fill out when creating new links.

When Ms. S created the link from the title of the timeline to the first frame of the map, she filled out a Link Property sheet, as shown in Fig. 4. In doing so, she entered descriptive text for each anchor and unchecked the Action check box associated with the anchor in the animated map document, thereby telling the system not to trigger the action associated with this anchor on link traversal.

What it means to 'execute an action' is application-dependent. If the application allows a range of possible actions, the application designer must decide what action to trigger when link following occurs. For example, when users follow a link to an anchor in an InterPlay document, we decided that 'execute an action' would mean play the animation from the frame in which the link anchor was created to the end of the animation.

> Bob opens the timeline document describing Napoleon's Campaign and traverses the link associated with 'Battle of Borodino'. This traversal opens the animated map and triggers the animated sequence starting at the beginning of the battle. The animation plays through to the end.

Alternately, we, as application designers could have decided to make link traversal to an active anchor in InterPlay to play the animation from its start, stopping the action in the frame in which the link anchor was created.

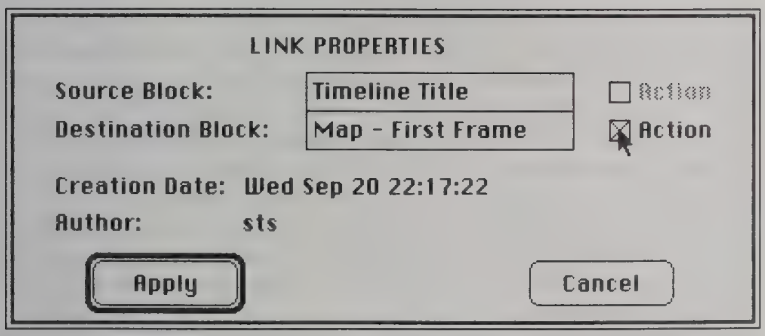


FIG. 4. *A dialog box that allows authors to enable or disable the action associated with an anchor (block). The 'Action' control associated with the destination 'Map – First Frame' is checked, indicating that the animation should execute when the link is followed to that anchor. The 'Action' control associated with the source 'Timeline Title' is dimmed and disabled, since the anchor is in a static document*

> After trying out a few links between the timeline and the map, Ms. S points out a possible problem. Following a link from a frame in the animated map to the timeline produces a straightforward effect: the timeline window is activated, and the appropriate date (the destination anchor of the link) is highlighted. However, every time she follows a link to the map, the animation plays forward to the end, leaving her at the same point (the return of Napoleon's skeletal army to Poland) no matter which link in the timeline she followed.

> She wants to be able to link each timeline event to only the corresponding segment of the army's journey, so that students can follow the invasion's progression in stages.

We may, in the future change what it means to 'execute an action' if we extend InterPlay's functionality to allow users to define spans of frames. In that case, we might play the animation starting from the frame in which the link anchor was created and halt playback when the end of a user-defined span is reached.

InterPlay is somewhat unique in that the application allows users to directly select dynamic information. The starting point of the action, therefore, can be determined by the point in time an anchor selection is made (i.e. the frame in which the link anchor is established).

In the case of applications that do not permit direct selection, Catlin recommends the use of a *proxy*[9]. For instance, musical sequences on an audio disk can be represented by entries in a table. Table entries, or proxies, can then be selected and made into active anchors. These entries can also serve to define the span of an anchor, as is done in our InterVideo application. Fig. 5 illustrates the use of proxies in the InterVideo application. Entries in the InterVideo tables include stills (single videodisc frames), clips (a range of videodisc frames), and sequences (an ordered collection of stills and clips). When a user follows a link to a still, a single frame is displayed. On the other hand, if a clip has been selected as a link anchor, the application plays a span of videodisc frames when the user traverses the link.

While we have done the majority of our experimentation on playback-style applications, we believe that it will be equally easy for developers of recording and query applications to define the meaning of 'execute an action.'

For recording applications, the developer might choose to turn on recording for a predefined time period or turn on recording and leave it on until recording input stops (as is the case with a telephone answering machine).

For query applications, a number of different types of actions might be possible. For example, a professor might want to create a link between a text document containing a bibliography of important works on a topic and include in it a link to an online bibliographic database application. When a student follows the link, the query that the professor formulated to search for all items on the topic would execute. This would ensure that each student received the most up-to-date information available at a given time. In such a case, the developer might choose to display the query specification upon a link traversal and then automatically run the query, presenting the results when the query is complete. Another option is to just run the query and

present the results without showing the query specification.

But a query may take several minutes to execute or an animation at the end of a link may be five minutes long.

— When users traverse a link to an active anchor, what options should users have *after* the destination document opens?

If the link to the active anchor does not trigger any action, then users should be able to perform all the functions possible after following a link to a static anchor. These include editing data, following existing links and creating new links. (In Intermedia, users may edit data or create links only if they have the appropriate access rights.)

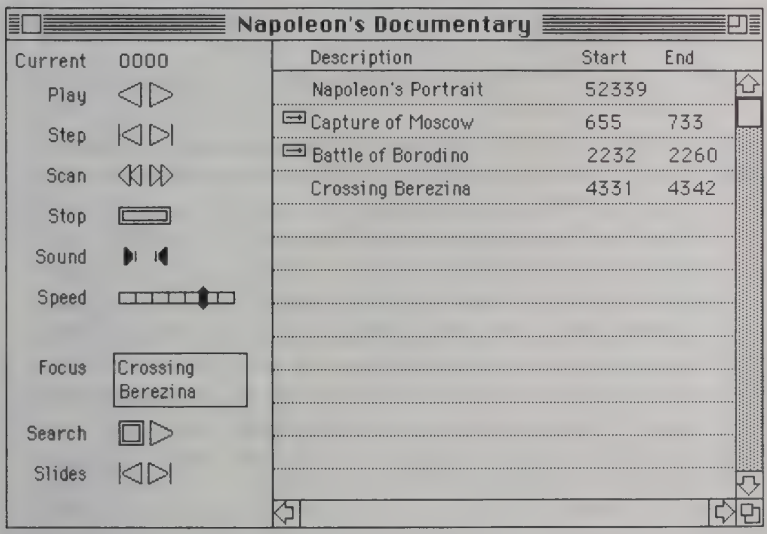


FIG. 5. *In the Intervideo application, spans are indicated in the table entries by frame numbers entered into the 'Start' and 'End' columns*

In addition, users should be able to control the action itself. This may mean running an animation, recording a voice message or issuing a query. Unlike following and creating links, which have the same interface across all Intermedia applications, users may have to know some application-specific details in order to interact with an action-oriented application. Since Intermedia is simultaneously an editing and a browsing environment, authors find that hiding tools and controls helps to keep the screen uncluttered for browsers. Consequently, authors often opt to hide these tools and controls once they are finished editing a document. This practice does not much

inconvenience users, since we have used a consistent set of menu commands across all applications for hiding and showing tool palettes and control panels. Even if a user is not familiar with a particular application, he or she should be able to find the tools and controls.

If users follow a link to an active anchor which does trigger an action, users will obviously want to watch the action, record data or wait for a query to execute. Since link traversal to an active anchor may initiate an action that could take anywhere from a few seconds to a few hours, it is inevitable that users will want to terminate actions from time to time. This is particularly true if the duration of an action is long, or if the user follows a link to the same active anchor numerous times in a session and does not want to wait for the action to run its course each time.

— When and how should users be able to stop actions at the destinations of links?

Sometimes users can anticipate that they do not want to wait for actions to happen, no matter how short or long they are.

> When Ms. S is proofreading text in all her different documents, she does not want the animation to start up each time she follows a link to the map. On the other hand, she does not want to change the action setting on all the anchors in the map which she has so carefully set up for her students.

Intermedia does not currently have a mechanism for users to temporarily 'turn off' actions, but this seems like a feature that users would want. If we were to add such a feature, users might have the option of temporarily disabling the action setting on all anchors for the current session. For example we might include a choice in a dialog box that allows users to check an 'All', 'None' or 'As Specified by Author' option. 'All' would temporarily turn on the action setting for all anchors, 'None' would temporarily turn off the action settings, and 'As Specified by Author' would leave the settings as they were set by the author of each link. Alternately, users may be able to only turn off the action setting for all links to a particular action-oriented document, such as the animated map. A third possibility is for users to specify that they do not want an action to execute each time a link is followed. This last possibility is the least appealing, since users would have to know whether or not the destination anchor was an active anchor and would also have to remember to follow the link in a different manner (for example double-clicking on the marker with the Shift key held down rather than just double-clicking).

Even though we do not have a feature for stopping actions before they begin, we do have a built-in way for users to stop actions once they

have started. Although most action-oriented applications have a mechanism for stopping actions, users can not always be expected to know what that mechanism is. In the course of following links, users may travel to unfamiliar territory, traversing a link to a document of a type that they have not encountered before. We therefore decided that Intermedia must provide an application-independent mechanism for stopping actions, one over and above the application-specific mechanisms that may already exist. For example, HyperCard users can type '⌘.' to terminate an action. In Intermedia, we allow users to press any key on the keyboard to terminate an action.

> After traversing the link from the 'Battle of Borodino' entry in the timeline document, Bob sees the animated sequence starting from the beginning of the battle. When the army in the animation moves to the next battle, he presses the space bar to halt the action.

Once an action terminates on its own or is halted by the user, users should have the same set of options as described above. They should be able to edit the content of the destination document, follow existing links, create new links or interact with the action-oriented application.

> While he was watching the 'Battle of Borodino' animation, Bob notices that when the army moves, the link marker associated with it moves as well. When the animation stops, the link marker and the contents of the anchor (the symbol representing the army) are both highlighted.

— If the content of an anchor changes over time, what should happen to the marker associated with the active anchor?

This issue only arises in a subset of action-oriented applications. InterPlay is one such example as is demonstrated in the scenario above. Objects that comprise link anchors may move over time or may even disappear during the course of an animation. Another application where the content of an anchor may change over time is an application that renders three-dimensional objects (Swivel 3D or a CAD package). If we assume that markers remain in a fixed position relative to the objects they are associated with, it would be possible for a user to rotate an object and obscure the marker behind the object. It would also be possible for the user to rotate a set of objects so that both the object that comprises the anchor and the corresponding marker are hidden behind other objects.

In both these cases, the application designer could easily come up with a scheme that would prevent markers from being hidden from view. In InterPlay, markers could remain visible even when their

extent is hidden. In a three-dimensional application, markers could always be displayed on the front-most plane. Although Intermedia allows each application developer to make the decisions about handling markers while their application is 'active', we believe that moving a marker, or hiding it when its corresponding extent is hidden, helps users to perceive the coupling of the marker and the objects that comprise the anchor.

Another mechanism to aid users in figuring out which objects are associated with which markers is highlighting. In static Intermedia applications, when a user follows a link, the content of the destination anchor is highlighted in addition to the corresponding marker. The rationale for this behavior is to focus users' attention on the destination of the link.

> Sue sees that there are two links associated with the 'Battle of Borodino' entry on the timeline. She has already followed the link to the animated map and now she would like some additional information about Borodino. The second link takes her to a lengthy essay on Napoleon's Russian campaign. When the essay opens, it is automatically scrolled about a quarter of the way through the essay. The three paragraphs about the Battle of Borodino are highlighted, as is the marker in the upper left corner of the first paragraph.

In Intermedia, highlighting the content of static anchors uses the same mechanism as selection, only slightly altered. For example, in text documents, blocks of text are highlighted using reverse video when the user creates a selection region. When an anchor is highlighted, the block of text is surrounded by a marquee. The situation is substantially different when considering anchors within action-oriented applications. In most applications there is no concept of selection while an action is executing; however, that is the time when the user's attention needs to be focused on the content of the anchor.

— How can the content of active anchors be highlighted?

Once again, the resolution of this issue is application dependent. For applications that use proxies, there is really no problem. The proxy can be highlighted while the action is executing. This gives users somewhat indirect feedback, but it does not, in any way, interrupt or obscure the action. For example, when a user follows a link to an InterVideo sequence, the name of the sequence is highlighted in the sequence portion of the InterVideo table while the stills and clips that make up the sequence play on the video monitor.

Query type applications are also straightforward since the data typically does not change once it has been retrieved. If the application designer has decided to display the query specification, then this can be highlighted along with the results. If not, just the results can be highlighted.

InterPlay poses a somewhat different problem. If the objects within an anchor were highlighted during the animation, for example, using grey selection handles, the animation would look different from what the author created. Our solution was to use the grey selection handles at the beginning and at the end of the animation. Before the animation begins, the user briefly sees the contents of the destination anchor. When the animation finishes, these objects are again highlighted. This solution is somewhat of a compromise, but we felt it would be distracting and not aesthetically pleasing to have any sort of highlighting occurring during the animation.

This issue of focusing the user's attention is complicated even further when considering the possibility of following more than one link at a time. In our future work, we plan to experiment with mechanisms that will allow authors to specify that traversing a link would bring the user to multiple destinations.

— Assuming that a link can have multiple destinations, how should the simultaneous playing of active anchors be co-ordinated?

If users can follow more than one link at a time, it is natural for users to want to employ several media in unison. For example, if Prof. K has a videodisc documentary on Napoleon and a CD recording of Tchaikovsky's *1812 Overture,* he might want to create a link to a videodisc sequence that shows a series of paintings done of Napoleon's retreat. While the sequence is playing, he would like a portion of the *1812 Overture* to play simultaneously.

Some existing applications, such as MacroMind's Director, have built-in synchronisation functionality. While this is useful for users of Director, it is not a general-purpose facility that helps users synchronise data from any application on their desktop.

To support the simultaneous activation of different anchors in the generic case, we need to consider a model that has one or more anchors at either end of the link. In effect, instead of being 'single-anchored,' links would be '*n*-anchored,' where *n* is the number of anchors at the destination end of the link. Traversing a link from a single source anchor to multiple destination anchors would activate whatever documents contained the anchors, and initiate whatever activity is associated with those anchors.

Since today's windowing systems only permit a single window to be

active at a time, we envisage that an *n*-anchored link would open (or activate) the windows and highlight each anchor in turn, leaving the last one highlighted in the active window. After this sequence is finished, the actions associated with the anchors would be triggered together, and proceed to their predefined endpoints.

One aspect of activating multiple anchors that is likely to be significant to users is the precise synchronisation of events. For example, the user may well wish to have an audio track begin playing at exactly the same moment as an animation sequence begins its motion. Although this is a significant issue, it is highly dependent on the technologies of the devices involved. In practice, until the technology advances, users might have to live with whatever slippage they encounter in trying to precisely synchronise active anchors. Since the response time on a computer may be subject to numerous variables (such as the current user load on a multi-user system), it may not be possible to reach a definitive solution.

One possible user interface that we have been contemplating is the introduction of two controls, 'Create Yoke' and 'Unyoke' in Intermedia's Link Properties sheet, as shown in Fig. 6. In the figure, the source anchor *June 23* is linked to three single-anchors at the destination: *Departure from Kowno, Battle Sequence,* and *Tchaikovsky's Excerpt.* In this interface model, anchors that have been joined together with a yoke are activated together when the link is traversed. In our current implementation, users would have to follow

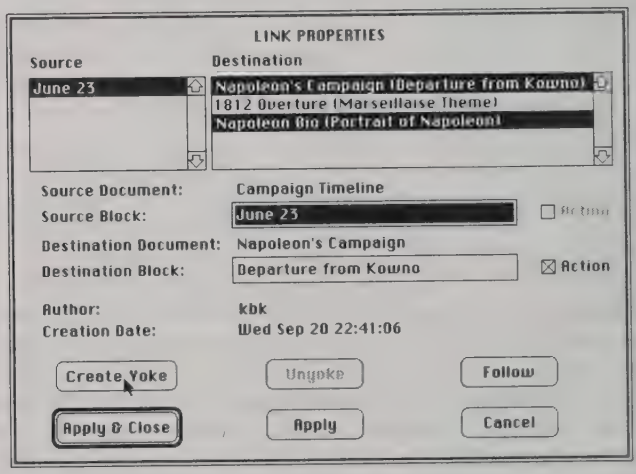


FIG. 6. *Proposed dialog box for creating yoked-links and unyoking yoked-links. The 'Unyoke' control is dimmed since there are no yoked-links.*

these links one at a time. To create a multiply-yoked link attached to the 'June 23' anchor so that link traversal from this anchor would activate *Tchaikovsky's Excerpt* and *Departure from Kowno,* a link author would select both anchors and click on 'Create Yoke.' The source anchor now has one link with two destinations and one link with one destination. To remove an anchor from an '*n*-anchored' destination, a link author would select that anchor and click on 'Unyoke.'

> Using their hypothetically enhanced version of Intermedia, Prof. K and Ms. S construct the links they want between the timeline, animation, video, and audio documents.
>
> Prof. K selects the link marker on the timeline for Napoleon's departure, and chooses the 'Start Link' command. He then selects the InterVideo frame reference for Napoleon's portrait, and chooses 'Complete Link.' Prof. K opens an InterAudio document, selects a sound bite of the *Tchaikovsky's Overture* and issues the 'Complete Link' command a second time. Finally he selects the animation sequence depicting the *Departure from Kowno* and makes a third link from the timeline. Next, he selects the link marker in the timeline and chooses the 'Link Properties' menu command; this command displays the dialog box shown in Fig. 6. Prof. K selects all three anchors at the destination in the Link Properties sheet and clicks on the 'Create Yoke' button. When he returns to the timeline and follows the link, the portrait appears on the video monitor, a snatch of Tchaikovsky's reworking of the 'Marseillaise' issues from the speakers of the CD player, and the army begins the first leg of its departure from Kowno.
>
> Ms. S goes on to create the rest of the desired links. She tests each one after creating it, following the link and checking the actions that occur. Since she is familiar with all the material by now, she often cuts short the playback by pressing any key on the keyboard. In the multiply-yoked link, Ms. S finds that pressing any key stops the playback of the video, audio, and animation sequences.

The last design issue to consider involves implementation.

— What modifications need to be made to the storage mechanism in order to accommodate the use of active anchors?

In Intermedia, storage of the link and anchor information is separate from the storage of the documents itself. The modifications

necessary to support active anchors represent an evolution in the life cycle of the storage mechanism, rather than a redesign, and reflect a corresponding evolution of functionality in hypermedia.

The data elements we regard as necessary are:

— an *action flag,* indicating whether the action associated with an anchor is to be executed;
— a *variable-length contingency string,* to be used by applications for any special-purpose action-oriented processing of the anchor. The existence of this data depends on the particulars of a specific application. For example, applications using the playback and recording paradigms may store their starting and ending frame numbers of action, whereas applications using the query paradigm may store the names of the database(s) in the order in which to perform the queries. The contingency string data is maintained by an application, and is totally invisible to the user, though the data may be the indirect result of a previous user interaction.

Such extensions to the storage mechanism constitute a path of outward mobility for hypermedia systems to merge new action-oriented applications into their environment.

4. CONCLUSIONS

In this paper we have classified action-oriented applications into three paradigms—playback, recording and query—to help readers think about the design issues involved in including action-oriented applications into a hypermedia environment.

In summary, we examined the following issues: When a user follows a link, what should happen when the destination document opens and what options should a user have at that point? If one user option is halting the execution of an action at the destination of a link, when and how should users be able to do this? What should happen to a marker associated with an anchor that moves or disappears, and how can that content be highlighted? If the system were to support multi-destination links, how can active anchors be coordinated? Finally, we addressed the issue of how storage mechanisms must be altered to accommodate active anchors.

Although the architecture of Intermedia is not identical to the architecture of other hypermedia systems, we believe the issues raised in the context of Intermedia have implications for other systems. Our experience leads us to suggest several rules of thumb.

First, it is useful for the hypermedia operating environment to provide a generic action flag so that authors of links can determine

whether or not an action should execute when a particular link is followed. Unlike using scripts to trigger actions, having a generic facility of this sort means that authors can turn actions on or off within the desktop environment, without any programming and without knowing the operational details of each application.

Second, hypermedia systems can encourage users to create links to dynamic information by providing a uniform interface for link creation across static and dynamic applications. This uniform interface should extend to the creation of multiply-yoked links (synchronised anchors) for applications that can run simultaneously.

Third, if there are many possible actions that can be triggered on link traversal, it will be helpful if application developers define the default one to be the most predictable and informative to the user, and the easiest for the author to define.

Fourth, developers must keep in mind that users may not be familiar with the operation of an application just because they have followed a link to it. Applications should therefore be designed so that users who are just browsing can garner useful information without interacting with the application beyond performing actions that are generic to the current operating environment (e.g., following another link, closing the document, or stopping an action).

Fifth, if there is any ambiguity about the destination of a link, the application developer should include some mechanism to draw users, attention to the exact destination the author of a link specified when the link was created. This mechanism should be clear, but should interfere as little as possible with the presentation of the information itself.

We offer these rules of thumb as a starting point. As new multimedia applications are invented, these rules will have to evolve and change so that new classes of applications can be effectively integrated into the world of linked information.

5. ACKNOWLEDGEMENTS

The concepts expressed in this paper were refined with the gratefully acknowledged assistance of Karen Smith Catlin, and of the Action Links Committee, which also included Bernard Haan and Jim Grandy. Special thanks to David Temkin for his editing and to Helen DeAndrade for her help in preparing the figures for this paper. This work was sponsored through a research contract with Apple Computer.

AUTHORS'S NOTE

Correspondence to the authors should be sent c/o IRIS, Box 1946, Brown University, Providence, RI 02912, USA.

REFERENCES

1. AKSCYN, R.M., McCRACKEN, D.L. *and* YODER, E.A. KMS: A distributed hypermedia system for managing knowledge in organizations. *Communications of the ACM, 31* (7), 1988, 820-835.
2. HALASZ, F.G., MORAN, T.P. *and* TRIGG, R.H. NoteCards in a nutshell. *In:* J.M. Carroll *and* P. Tanner, *eds. SIGCHI + GI 1987 Proceedings.* Toronto: ACM, 1987, 45-52.
3. KAEHLER, C. *HyperCard power: techniques and scripts.* Reading, Mass.: Addison-Wesley, 1988.
4. GOODMAN, D. *HyperCard developer's guide.* New York: Bantam Books (Macintosh Performance Library), 1988, 564, 573-577.
5. YANKELOVICH, N., HAAN, B.J., MEYROWITZ, N. *and* DRUCKER, S.M. Intermedia: the concept and the construction of a seamless information environment. *IEEE Computer, 21* (1), 1988, 81-96.
6. CONKLIN, J. Hypertext: An Introduction and Survey. *IEEE Computer, 20* (9), 1987, 17-41.
7. APPLE COMPUTER, INC. *Inside Macintosh.* Reading, Mass.: Addison-Wesley, 1985.
8. GOLDBERG, A. *Smalltalk-80: the interactive programming environment.* Reading, Mass.: Addison-Wesley, 1984.
9. CATLIN, T.J.O. *and* SMITH, K.E. Anchors for shifting tides: designing a 'seaworthy' hypermedia system. *In: Online Information 88: Proceedings of the 12th International Online Meeting.* London: Learned Information, 1988, 15-25.

K. PARSAYE, M. CHIGNELL, S. KHOSHAFIAN *and* H. WONG. Intelligent databases: object-oriented, deductive and hypermedia technologies. New York: Wiley 1989. 479pp. £20.15. ISBN 0 471 50345 2.

Intelligent databases are *databases that manage information in a natural way, making that information easy to store, access, and use.* Such tools are essential if the ever increasing information overload is to be handled efficiently. The authors of this book set out to provide a blueprint of how such intelligent databases might become a reality in the foreseeable future by developing an architecture based on what they regard as the five crucial enabling technologies: traditional databases; object orientation; expert systems; hypermedia; and text management/information retrieval. Before presenting their unifying model, each of the above topics is discussed in a chapter of its own in such a way that the novice has a reasonable chance of grasping the key concepts involved.

Chapter 2 discusses the traditional approach to database design. The emphasis is primarily on relational databases, providing a very readable description of, for example, the relational algebra, query languages, design methodologies, and Codd's (somewhat controversial) thirteen rules defining the criteria for what makes a database relational.

Chapter 3 provides an introduction to object orientation, undoubtedly the most popular programming paradigm around at the moment. As we find out in chapter 7, object orientation turns out to be the 'glue' which holds together the authors' proposed intelligent database. This would certainly explain why this chapter in particular seems to have been carefully thought out and composed — in my opinion, the best chapter of the book. Even prototype systems and delegation are given a mention.

Chapter 4 describes expert systems and their importance primarily with respect to traditional and object oriented databases. The authors' main view of expert systems in this context is as means of employing more sophisticated programming techniques.

Chapter 5 discusses hypermedia. Again the treatment is thorough — the history, rationale and commercial availability of hypermedia is presented. The conceptual similarities between hypermedia and the AI knowledge representation formalism of semantic nets is noted. Specific implementations that are discussed include HyperCard, NoteCards and Guide. Interestingly, of the five, hypermedia is the technology that is least integrated with any of the others.

The last building block is presented in chapter 6 — text management and information retrieval. The importance of this technology is that it allows information of a textual nature to be incorporated into the database, the other four technologies do not provide an effective means of handling such implicitly structured information. Specific topics covered in this chapter include: automatic indexing; query formulation; relevance feedback; performance metrics; concept indexing; and hierarchical indexing.

The core of the book is chapter 7 which describes FORM (Formal Object Representation Model), a general model of intelligent databases built from the five technologies discussed in chapters 2-6. The model consists of three layers. The first layer comprises a number of high level tools providing facilities such as intelligent search and integrity control. The second layer, the high-level user interface, incorporates a hypermedia object-oriented representation of information with an accompanying set of tools for browsing, searching and querying. The final layer is the intelligent database engine which utilises a deductive object-oriented representation of information.

At first, the thought of constructing an intelligent database based on the unification of five different paradigms for information management seems a bit of a 'Holy Grail'. However, having read the book, I now admit to being more convinced that such a system is at least feasible, although probably not in the near future. Regardless of that, the book certainly provides an excellent review of the five key information technologies, and more importantly, the relationships and overlaps between them.

DAVID G. KARETNYK
Department of Information Science, University of Strathclyde

NOEL WILLIAMS *and* PATRIK HOLT, *eds. Computers and Writing.* Oxford: Blackwell Scientific Publications, 1989. £19.95. ISBN 1 871516 03 X.

The process of writing is so necessary and yet often so time-consuming and tedious that there is, it seems, a continuing urge to subtract the pain: to make writing easier, faster and more effective. The advent of word-processing has clearly made easier the creation and editing of text, its revision and manipulation. Yet there is still a drive to further involve computers at every level of the writing process.

The book under review comprises a collection of papers — most of them rewritten for publication — from the first conference on computers and writing, convened by the Communication and Information Research Group (CIRG) at Sheffield City Polytechnic in April 1988.

The importance of the relationship between computers and writing, suggests Noel Williams, has been recognised for longer in the United States than it has here. In this country that relationship tends perhaps to be seen within a range of contexts — such as Computer Assisted Learning, software documentation and natural language processing — in none of which is the focus specifically on 'the value of computers for writers, the use of the computers to gain or develop writing skills or the effect of computers on human writing processes'.

The Sheffield conference sought to address these points by bringing together a number of speakers from different fields and among the questions considered are: 'Designing computational writing tools within a linguistic model of the writing process' 'Computational story telling', 'Developing a writer's assistant' and 'Guidelines for multiple users creating hypertext: SQL and HyperCard experiments'.

Williams, in his own chapter 'Approaches to computer assisted writing', discusses the characteristics of good writing software. These depend, of course, on the needs of the users, and for these purposes he divides software into writing *support* and writing *training* tools. Broadly speaking, support tools are needed by professional writers, by those like academic writers and journalists who are concerned to improve what they themselves write; training tools are necessary to those who teach or are learning how to write, or how to write in a specific way.

Clearly each group has a number of different requirements. For instance, support material will give automatic corrections and suggestions without offering intrusive comments or detailed information, while training software will be well supported in documentation and will not allow the user to implement automatic correction. In view of this, it is necessary to identify broad categories of potential users and needs when designing writing software, and much of this chapter is devoted to a description of *Ruskin*, a postwriting system developed at CIRG for IBM PCs.

Ruskin is designed to teach report writing and includes the contextual variables of audience, purpose and subject matter combined with a number of broad categories of feature such as length of structures, proofreading checks, word lists and presentation. The system seems useful, and the discussion in the chapter is interesting and important.

In their chapter, Sanford and Moxey consider some problems of writer's style-aids. These vary in range and in sophistication from simple spellcheckers to software offering stylistic advice. The general goal behind research on computer style-checking, they say, is 'to provide the writer with the means to improve the style and presentation of the final product' and the chapter examines some of the criteria for design of a 'useful and usable style-checker'.

The writers discuss some style indicators and the problems of dealing with these by computer. Spelling errors and ungrammatical sentences, which are perhaps more questions of accuracy than of style, they see as easier to deal with by computer than — a more 'stylistic' example — sentence length, although spelling and grammar pose their own problems.

But in punctuation, which relates both to grammar and to style, problems persist. The comma, for instance, may be used to indicate a pause in the reading of a text, to avoid ambiguity or to establish rhythm in a sentence. It can also be affected by personal taste and literary fashion. And since its uses are so varied, programming for its insertion or deletion is correspondingly difficult.

On sentence length and modulation, the authors suggest that little research has been done in this area and that it is difficult to conceive of computer-delivered advice much more specific than, for example, 'Write more short sentences'. They go on, in a way that they describe as 'quite speculative', to suggest that a solution may be found in a variation of the traditional advice to 'Read over what you've written' — but with the use of sound. They look to research into technology which will enable a writer to play back and listen to what he has written and thus detect stylistic flaws.

This is an interesting chapter, amusing in places, which certainly states rather than solves problems. To turn from it to Unni Hovstad's chapter on computer-assisted essay writing is to leave the world of speculation for the practical workaday world of sixth-form language teaching.

Hovstad gives an account of a project conducted in a Norwegian grammar school using computers in a variety of ways to teach English as a foreign language. Among the aims of the project were to introduce arts students to the computer as a relevant tool for essay-writing, to reduce poor usage and to stimulate students to increase their linguistic skills.

Though complicated by insufficient time and shortage of equipment, the project — which seems to be continuing — has been judged successful. Among the computer features utilised have been: mistake boxes; count checkers of words and word lengths; and

checkers of the numbers and lengths of sentences and paragraphs. To review syntactic misconstructions, printouts are used of sentences longer than a student's average, while printouts of the first and last sentences of each paragraph ensure adherence to the essay's theme and give the opportunity for study of paragraph linkage.

While such a programme is not one to set the blood racing, it will be judged by its effectiveness. And it would be useful to see it used in teaching the mother tongue and, in a restricted way, in remedial education.

Computers and Writing offers, then, a variety of topics and approaches within an overall theme. The enthusiasm of the contributors is apparent and two outcomes of the Sheffield conference are the decision to hold an annual conference or workshop and to publish a newsletter devoted to computers and writing. Both ventures should prove successful.

DANIEL BAIRD
Strathclyde Regional Council Education Department

TERRY WINOGRAD *and* FERNANDO FLORES. Understanding computers and cognition. Reading MA: Addison Wesley, 1988. 207pp. $12.95. ISBN 0 201 11297 3.

An initial glance at a book with such a title might lead one to imagine that this is just another text on human-computer interaction (HCI) or a basic reader in cognitive science, but both inferences would be false. Quotes on the cover from such notable reviewers as Joseph Weizenbaum and Howard Gardner testify to the book's 'ground-breaking', and 'deeply thought-out' qualities which render it a 'very good and important work'. The publisher's blurb makes numerous references to its 'controversial' 'new', and 'broad-ranging' contents before summing it up with the statement that the book enables us to better understand 'what it means to be human'! Who wouldn't be intrigued?

This book is sub-titled 'A new foundation for design' and herein lie the clues to its contents and purpose. Winograd and Flores are interested in computers as life enrichers and liberators of humanity, and encourage designers to 'let our awareness of the potentials for transformation guide our actions in creating and applying technology'. (p.179). However, such admirable goals are, according to the authors, unattainable within the design philosophy underlying contemporary technology which is presented here as narrow and

rationalistic. What the authors attempt to provide is a clarification of the background of understanding in which the discourse about computers and technology takes place. In so doing they lead the reader on a mind expanding trip through hermeneutics, the biology of cognition and speech act theory (all this in the first sixty pages). This occurs before what we might euphemistically describe as 'highlighting the shortcomings' of current work in AI and intelligent systems; they then put forward their alternative. As you will no doubt have noticed by now, this is no ordinary book.

The basic tenet of their argument is that computer science and psychology have got it wrong! The success of the hard sciences in identifying cause-effect relationships and building a body of knowledge about how the world works has led, inappropriately, to an acceptance of empiricism as the only way to further our knowledge of ourselves and our psychological world. Such an approach demands a view of humans as existing in, but distinct from, environments filled with objects on which we perform operations. In the opinion of the authors, this has resulted in generations of researchers and system developers taking for granted a false view of how we exist in and interact with our surroundings, with the end result that our computerised tools are a poor match for our needs. According to the dominant view, the human cognitive system is to be unquestioningly accepted as an information processing system capable of representing the world symbolically, and intelligent behaviour is said to result from the correct manipulation of these representations in the mind. Not so, claim our intrepid explorers of uncharted intellectual territory; this is 'the fallacy of instructive interaction'. In so saying, they dismiss thirty years of work in cognitive psychology.

Their alternative is based largely on the work of Heidegger in that it claims objective perception and detached analysis of the world are fundamentally impossible. There is no knowledge without involvement; 'being-in-the-world' (*Dasein,* in Heidegger-speak) and 'throwness', which they define as 'the condition of understanding in which our actions find some resonance or effectiveness in the world' (p.33). In support they call on the work of Maturana and Searle. The former unknown to me is concerned with the biology of cognition and argues strongly in favour of 'structural coupling' between organism and environment. Concepts such as 'autopoiesis' and 'consensual domains', are explained but fall short of providing me with enlightenment. Searle is invoked in order to highlight the limitations of formal analyses of language that fail to consider the role of background information and the necessity of commitment to certain views in understanding the meaning of a statement.

The upshot of all this is the initially surprising proposal that 'nothing exists except through language' and therefore:

> 'the domain in which people need to understand the operation of computers goes beyond the physical composition of their parts, into areas of structure and behaviour for which naive views of objects and properties are clearly inadequate. The 'things' that make up 'software', 'interfaces' and 'user interactions' are clear examples of entities whose existence and properties are generated in the language and commitment of those who build and discuss them' (p.69).

Attempts at building expert systems or natural language processors based on the dominant rationalistic model are doomed to failure according to the authors and they expend several chapters of effort describing and destroying some of the ludicrous claims made for computers and applications like ELIZA and SHRDLU. So far so good.

The final section of the book attempts to lay the foundations for a new design philosophy based on the earlier discussions. Unhappy with the perspective that views users of technology as rational decision makers searching a problem space considering alternatives, they present the user as 'thrown' into a problem situation, possessing a set of 'background assumptions' about the situation he is in and having 'commitment' to the attainment of certain objectives. From this viewpoint, organisations are seen as 'networks of commitments and recurrent conversations' in which 'breakdowns' occur to which members of an organisation respond. The role of the tool is to support communication between members; it is a means not an end; and to be successful tools must be transparent.

The ideas are not difficult to grasp and indeed, some are not particularly original. What is required at this point is a description of a system embodying such ideas. What we get instead are references to, but few details of, something called 'The Coordinator Workgroup Productivity System', a claim that the Apple Macintosh supports the 'readiness-to-hand', 'ontological simplicity' of their approach and some vague guidelines grouped misleadingly under the title 'A design example'. If you want to know how to do it, you won't find the answer here.

In fairness to Winograd and Flores they admit that they 'offer no magic solutions, but an orientation that leads to asking significant questions' (p.174). In this sense, the book is a runaway success. The authors adopt a position that some might see as scientific heresy and interestingly (and, to my mind, successfully) defend it against the imagined attacks of 'rationalists'. This is done in a manner and form that few will find too difficult to follow. It would have been all too easy to hide behind the often impenetrable prose of hermeneutics and

linguistics but the authors, to their credit, have selected most of their terms carefully and argued clearly.

My criticisms are that they occasionally set the 'rationalist' position up as a straw man, caricaturing the dominant view in a form to which few would subscribe. For example, the concept of theory-free observation has fewer supporters in cognitive science than Winograd and Flores might suppose. They also fail to provide a genuine example of how a system designed according to their approach might look. To those of us concerned with designing systems for real users, this is a shortcoming that no amount of intellectual posturing can redress. For example, on the subject of electronic libraries, few of us interested in hypermedia will take much inspiration for design from the conclusion that 'in providing a tool (for browsing and searching material) we will change the nature of how people use the library and the materials within it' (p.167).

Notwithstanding this is an interesting, occasionally controversial and in some senses, ground-breaking book. (Who am I to argue with Weizenbaum?). Readers with a background in linguistics and/or hermeneutics might wonder what the fuss is all about; they have known all this for years! Others who have been dismayed by the manner in which technology and design are implemented might be relieved to see written expression of their thoughts reach such exalted heights, and intrigued that there is an established philosophical thesis they can adopt to lend credence to their position. Some people might even be surprised. Thus it is the sort of book many will find either stimulating or totally irrelevant. The ideas here are not new *per se,* but they are unorthodox in the domain of HCI and if for no other reason than this, it must be recommended.

ANDREW DILLON
HUSAT Research Institute, Loughborough

Online Information 89. Proceedings of the 13th International Online Information Meeting, London, 12-14 December 1989. Oxford: Learned Information, 1989. 609pp. £70.00.
ISBN 0 904933 72 5.

It was, on the face of it, business as usual at last December's International Online Meeting at Olympia. An ever-growing number of stands and exhibits bore witness once again to an astonishing proliferation of new products, services and media; proof, were it needed, of a seemingly relentless supply-side drive from the industry.

And, while it was difficult not to be impressed by the scale of the exhibition, I did leave Olympia with a mildly sinking feeling, not wholly explained by the hole in my wallet after buying sandwiches and a few drinks!

It's often remarked by far wiser commentators than the present writer that the next two or three years will come to be seen as a watershed for many electronic publishers. In business terms at least, the key to a viable online sector hinges on the lowering of costs which can only be achieved through scale economies. In other words, suppliers need to attract much higher volumes of service use by broadening their consumer base — which means hitting end-users, and hitting them hard. Percival and MacMorrow underline the point neatly in their paper in the published Proceedings on creating graphical interfaces to online services using HyperCard (p.276):

> '. . . we must, however, recognise the fact that information is a resource which is not at present being effectively exploited. Therefore anything which aids this exploitation must be welcome, as it will lead to expansion of the market, and therefore more services from more vendors, accessible to more people. All of which will result in healthy growth of the online industries, which will benefit everyone'.

Simply looking around the exhibition, I wasn't altogether convinced that the barriers to the development of such a mass consumer market are yet down, or even showing many signs of wear and tear. Since my own professional interests are very much concerned with issues of information market development, I turned to the Conference with these thoughts in mind. The Proceedings are now published by Learned Information and, as always, contain an exceptionally wide range of interesting material.

A very brief analysis of the content of this year's Proceedings offers two possibly rather obvious, but nonetheless interesting, conclusions. First, a surprisingly high proportion, nearly three-quarters, of the papers accepted for the Meeting came from practitioners and researchers from academic or other publicly-funded institutions. As usual, the Proceedings contain a wide range of contributions mixing papers on basic research in information retrieval; user studies; industry surveys; and reports on new applications. But, this year, with the notable exception of an excellent review of optical multimedia technologies which focuses on CD-ROM XA, by Ghislandi and Campara (pp.211-226), relatively few papers are concerned solely with technology. Instead, many are concerned with the development of local applications and with the fine-tuning of information services to user communities. A quick analysis of the 57 papers accepted for publication leads to the second conclusion that almost half the

contributions came from information mediators: from librarians, information scientists and system designers. They report extensively throughout the Proceedings on their attempts to tailor information systems and products to their own, peculiar local requirements. So, you will find papers which deal with the integration of external online services with inhouse databases (and CD-ROMs); with the development of local interfaces to a variety of data sources; and with providing the means for ordinary users to access often complex information structures through hypermedia.

If the Proceedings are to be regarded in any way as an indicator of current research directions, I, for one, am greatly encouraged. First, because there are clear signs that information mediators are indeed beginning to redefine their roles, and are seeking to reach and involve users in most direct — and less patronising — ways. To an extent, these developments flow from a general weakness in information product design and specification. Reading through the Proceedings, there are many examples of intermediaries customising and tailoring *generic* information products to the needs of their own clients. Certainly, there has been a significant shift in recent years towards the *decentralisation* in the development and enhancement of information systems. One of the contributors, Walter Claasen, in a useful discussion of information analysis in the humanities, makes a point (p.277) which can be authenticated simply by browsing the contents pages of the Proceedings:

> '. . . next to the developers of sophisticated information systems, such as those who have traditionally presented their information tools at the Online conferences, we will in future have *a large number of smaller developers* (e.g. institutions on a local level and inspired individuals) who will develop 'systems' through the use of (amongst others) hypertext'.

With some of these thoughts in mind, I want to highlight some of the more interesting papers which fall under three general headings, and address issues which are surely of critical importance for the healthy development of the online industries.

Encouraging end-use

Most encouraging of all are signs of a renewed interest in the pursuit (and encouragement) of end-users, an area where Anne Clyde's award-winning paper on online information services in Australian schools (pp.285-300) stands out. The contribution by Judith Copler (pp.549-555) discussed an exciting joint venture between Indiana University, Purdue University, the Department of Education and IBM which is currently targeting electronic information services at new groups of

potential end-users in selected schools and businesses; while Ray Thomas reviews progress towards the use of Computer Mediated Communication (CMC) techniques in adult education at the Open University (pp.303-319).

Mixed information environments

Another area where exciting developments are taking place is in the quest for *transparency* and *integration* in mixed information environments. New information storage and delivery technologies are gradually supplementing (and displacing) online access — but the complex mix of inhouse databases, CD-ROMs, hypermedia products, integrated library and document delivery systems is creating new problems — a multiplicity of access points. Regaining control of information delivery is the topic of three particularly interesting papers:

Wolfgang Donner (pp.71-75) gives a progress report on a project at Bayer AG to develop fully integrated access both to public and inhouse chemical databases, using the company's own RESY retrieval software. Bayer intends that end-users will soon be able to create their own databases; build corporate files such as bibliographies; search internal databases (such as chemicals on stock at Bayer); and search public as well as private files. Other contributions are concerned with the merging of online library catalogues and CD-ROM periodical indexes (Terry Webb, pp.363-372); and the combined use of internal databases, CD-ROMs and online services at the International Agency for Research on Cancer (Miido, pp.349-362).

Better interfaces

The quality of the interface between software and user is one of the most problematic issues in information system design. New facilities and enhancements are constantly being developed by information system suppliers and developers in an attempt to forge closer links between product/service design and market acceptance.

This theme is addressed by several papers in the Proceedings which direct themselves to the design and implementation of better interfaces. The work of a team at Queen's University (Smith et al., pp.1-12) prototypes voice access to an online service (BLAISE) and concludes that voice, the most natural form of communication, may be at least as fast as retrieval via a keyboard. MacMorrow and Baird (pp.13-22) describe some of the authoring tools available for use with Apple Computer's HyperCard which permit voice to be integrated into multimedia applications. Other authors look at the potential for

expert systems to enhance search effectiveness (Morris, Tseng and Walton, pp.415-434) or propose new paradigms for the design of hypertext documents (Simpson, pp.241-255).

For some reviewers, December's International Online Meeting proved to be rather disappointing — no revolutionary new searching gizzmos, no fundamental advances in information retrieval theory. Maybe, but a period for reflection, for consolidation and, perhaps, for reassurance was long overdue.

IAN ROWLANDS
Centre for Information and Communication Studies
Polytechnic of Central London

HYPERMEDIA

Hypermedia is an international journal designed to provide a focus for research and a source of information on the practical and theoretical developments in hypermedia, hypertext and related technologies. These highly innovative systems allow the integration of images, sound, text and data to form powerful tools for information retrieval; by linking multi-media with mass storage they can provide users with a more diverse and richly textured information environment.

The journal **Hypermedia** covers the following key themes: the conceptual basis of hypertext systems; cognitive aspects; design strategies; knowledge representation; link dynamics; authoring; navigation and browsing; testing and evaluation; user interfaces; tools for hypermedia; hypertext and expert systems; applications in education and training, information management, publishing, business, commerce, the professions and public administration.

The Editors will be pleased to consider for publication original manuscripts which deal with any of this broad range of themes. Papers should not have been published previously, or submitted elsewhere simultaneously. Papers presented at conferences may be considered if they are unlikely to be published in a conference proceedings volume.

Directions to authors

1) Two paper copies of the manuscript should be submitted, typed with double spacing on one side of the paper, and should be between 3,000 and 7,000 words. Pages should be numbered consecutively. The language of publication is English. Where possible an electronic version should also be supplied, on 3.5 disk in MacWrite, Word, WordPerfect or ASCII format.

2) The title of the paper should be informative. The name, designation, and affiliation of the author(s) should be given, and an abstract of 100-200 words outlining the scope of the paper and its main results or conclusions.

3) Headings (and sub-headings as necessary) should be included where appropriate).

4) Tables and diagrams should be placed on separate sheets, with their location noted in the text. Diagrams should be drawn in black ink, and be suitable for reproduction without re-drawing.

5) Italics may be indicated by underlining of the relevant text.

6) All acronyms or abbreviations should be spelt out in full when first used in the text.

7) Footnotes should be kept to a minimum, and avoided completely if possible.

8) Citations should be numbered in consecutive order, and included in the text in parentheses, as follows: 'It is clear that the points made by Jones (16) are of importance to all translators . . .' All citations should be listed in full at the end of the paper, in the same consecutive order. Multiple authors should be cited in full (not as *et al*).

Journal articles:
Citations should include the title of the journal in full, volume and issue number, and page numbers, as shown below:

NARIN, F. Interrelations of scientific journals. *Journal of the American Society for Information Science, 23(5)* 1972, 323-31.

Monographs:

KUHN, T.S. *The structure of scientific revolutions.* Chicago: University of Chicago Press, 1962.

Paper in conference proceedings volume, etc.

BELKIN, N.J. The problem of 'matching' in information retrieval. *In:* O. Harbo and L. Kajberg, *eds. Theory and applications of information research.* Proceedings of the Second International Research Forum in Information Science. London: Mansell, 1980, 187-97.

Research report:

JALLOQ, M.C. *Use of patent literature by academics in chemistry and physics, electrical and mechanical engineering.* London: British Library, 1982. (British Library Research and Development Report No. 5770).

9) Where necessary, authors should ensure that permission to reproduce material quoted in their papers has been obtained from the relevant copyright holders. The permission of any employer, supervisor or funding body should also be secured by the author if necessary, before submission of the paper.

10) Galley proofs will be supplied to the author or first author of a paper, and errors in typesetting should be noted, using guidelines supplied with the proof. No substantial changes in content can be made to the paper at this stage; the author(s) should therefore ensure that the paper is submitted in its final form.

11) Authors will receive a complimentary copy of the journal on publication, and the author or first author will also receive 20 complimentary offprints of the paper. Additional offprints are not available; permission to copy may be obtained from the Publishers.

12) The Publishers of the journal, hold joint copyright with the authors.

HYPERMEDIA

VOLUME 2 NUMBER 1 1990

ARTICLES

Increasing the power of hypertext search with relational queries
LEONARD GALLAGHER, RICHARD FURUTA *and* P. DAVID STOTTS 1

A hypermedia tutoring system for design trainers PHILIP J. GARTSHORE 15

Using an issue-based hypertext system to capture the software life-cycle process MARK LEASE, MAC LIVELY *and* JOHN LEGGETT 29

Linking active anchors: a stage in the evolution of hypermedia
MURUGAPPAN PALANIAPPAN, NICOLE YANKELOVICH *and* MARK SAWTELLE 47

REVIEWS

K. Parsaye, M. Chignell, S. Khoshafian *and* H. Wong. *Intelligent databases: object oriented, deductive and hypermedia technologies* DAVID G. KARETNYK 67

Noel Williams and Patrick Holt, *eds. Computers and writing* DANIEL BAIRD 68

Terry Winograd *and* Fernando Fiores. *Understanding computers and cognition* ANDREW DILLON 71

Online Information 89. Proceedings of the 13th International Online Information Meeting, London, 12-14 December 1989 IAN ROWLANDS 74